Escuela Politécnica Superior

Grado en Ingeniería de Sistemas Audiovisuales

Trabajo de fin de Grado

Desarrollo de aplicación móvil en Android para la gestión de competiciones de bolos

Autor: Alejandro Arjonilla García

Tutores: Carlos Alario Hoyos

Iria Manuela Estévez Ayres

Marzo de 2016

RESUMEN

Este trabajo se centra en el diseño y desarrollo de una aplicación en el sistema operativo Android. Ésta tratará de una aplicación de estadísticas sobre bolos americanos o en adelante "*bowling"* (*Sportsvite, 2016*). La aplicación desarrollada no es un juego, sino una herramienta pensada para el jugador que le permita hacer un seguimiento detallado sobre su juego, y de esta manera mejorarlo de acuerdo con las estadísticas conseguidas. El trabajo alberga todas las fases de desarrollo de una aplicación móvil, desde el estudio de las tecnologías adecuadas para su desarrollo, el diseño de la aplicación, su implementación y la evaluación de la misma.

Durante el estudio de las tecnologías se expone el uso de las mismas en la actualidad, y se analizan los pros y contras de desarrollar con unas u otras tecnologías la aplicación de este proyecto. Para el diseño de la aplicación, se estudiaron las posibilidades conforme a la tecnología elegida. Y de acuerdo con ello se hizo un diseño que más tarde pasará a ser desarrollado.

La evaluación de la aplicación, consta de dos partes: la evaluación funcional de la misma, comprobando que los requisitos funcionales tienen un comportamiento correcto de acuerdo con lo plasmado en el diseño funcional de la aplicación; y, además, se hace un pequeño análisis con un grupo de usuarios los cuales serán los usuarios finales, es decir los jugadores. Para ello, se les entregó un cuestionario en el que contestaron a una serie de preguntas tipo test, y tuvieron la oportunidad de sugerir mejoras, y de poder comentar cualquier tipo de fallo visto en la versión beta de la aplicación que han evaluado.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

En este capítulo se introducirá la motivación para el desarrollo de la presente aplicación, además de los objetivos que se quieren conseguir. Además, se explica el marco regulador que afecta a la aplicación en cuestión. Y en último lugar de este capítulo también se detalla la estructura seguida.

Hoy en día, las aplicaciones móviles forman parte de nuestra vida diaria. Utilizamos nuestros teléfonos móviles de manera constante a lo largo de nuestros días, y por ello, se desarrollan miles de aplicaciones diferentes en todos los posibles ámbitos. Por pequeño que sea el ámbito o el mercado, siempre aparece una aplicación dispuesta a cubrir esa necesidad. (PRESS 2016)

La aplicación que se desarrollará en los siguientes capítulos, se trata de una aplicación en Android orientada al almacenamiento de información en bases de datos, para su consiguiente estudio. Este estudio está dirigido al análisis de la información estadística de un jugador de bolos.

Para ello, se podrá hacer un estudio estadístico de las diferentes combinaciones, puntos conseguidos, porcentajes de las combinaciones, etc. Todo ello a sus diferentes escalas, series de juego, cómputo global, etc. Este estudio se podrá realizar mediante la visualización de los datos en listas con las estadísticas recogidas.

Esta aplicación tiene una base de datos en local, sin pasar por ningún servidor, dado que el objetivo fundamental es el almacenamiento de estas bases de datos de manera personal, para su consiguiente estudio y análisis.

Se eligió la plataforma Android por su estudio previo en la universidad, y por ser *Open-Source* de cara al pago de posibles tasas o permisos. A la vez la base de datos utilizada es la propia del sistema Android, la llamada SQLite.

A la hora de analizar los datos habrá diferentes opciones. Se podrán consultar los diferentes días (series) de juego, y dentro de esas series de juego, se podrán analizar los datos de esas series. Como los % de pleno, % de semi-pleno (derribar todos los bolos en dos tiradas), etc.

El objetivo de la aplicación es ofrecer al usuario un análisis de sus habilidades como jugador de manera sencilla, pero a la vez, con una amplia variedad de datos para poder hacer un estudio más profundo.

Por último, se expondrán las conclusiones que se han ido extrayendo en la realización de la aplicación.

1.1 MOTIVACIÓN

La motivación principal para la elaboración y elección de este proyecto, es el auge que en este momento hay por las aplicaciones móviles en el mercado. Desde el punto de vista personal, el desarrollar una aplicación completa, desde su diseño y desarrollo hasta su fase final, es una de las cosas que más me han motivado a la hora de elegir este proyecto.

Otra gran motivación para hacer este proyecto, es la necesidad de poder hacer una aplicación móvil para el ingreso de estadísticas para este deporte, para el cual, como en muchos otros, es muy necesaria la gestión de todas las estadísticas posibles. De cara a alcanzar la excelencia, estos detalles son de gran importancia para la mejora de la técnica de los jugadores.

En este momento ya hay aplicaciones en el mercado de las aplicaciones móviles (*My Bowling Scoreboard (Ho 2016), Bowling Scorer (Yotchan_lab 2016), Strike Out Stats (Bruno 2016))* la gran mayoría de pago, debido a que no hay un gran volumen de mercado, lo que provoca que tampoco haya mucha competencia entre las distintas aplicaciones. Este contexto sirve también como motivación para la creación como parte de este proyecto de una aplicación más sencilla de cara al usuario, pero con una utilidad plena.

1.2 OBJETIVOS

Los objetivos principales a la hora de hacer la aplicación son:

1.2.1 Definición de un plan de acción

En esta fase, se analizan las herramientas a utilizar para su desarrollo, con el objetivo de adquirir las habilidades en las tecnologías elegidas y herramientas añadidas de cara a su desarrollo. Entre las herramientas cabe destacar el uso de Git *(Git 2016)* para el control de versiones, o la utilización *mock ups.*

Esta fase incluye la adquisición de los conocimientos necesarios en software de desarrollo, como Android Studio, Eclipse, utilización de programas de control de versiones con su consiguiente elección, programas de diseño de aplicaciones móviles.

1.2.2 Diseño de la aplicación

En esta fase se plantea un diseño y hacer las consiguientes modificaciones de acuerdo a los cánones actuales de diseño de aplicaciones móviles. Se definen los requisitos funcionales de la aplicación, también se crea la arquitectura de la misma.

1.2.3 Desarrollo con las herramientas utilizadas

En resumen, el objetivo fundamental es la obtención de los conocimientos como si de una *startup* se tratara, tratando todos los aspectos fundamentales de desarrollo: toma de decisiones en función de la tecnología, caminos a seguir en función de unas necesidades, etc.

El trabajo consiste en la primera versión de BowlingStats, la cual cubre la gran parte de lo que podría ser una aplicación de estadísticas de *bowling*. BowlingStats ayudará a los jugadores de *bowling* a seguir mejorando en su juego a partir del análisis de sus estadísticas insertadas de manera fácil e intuitiva.

1.3 ESTRUCTURA DE LA MEMORIA

En el primer capítulo se hace una introducción al lector sobre el porqué y la necesidad de esta aplicación y se detallan los objetivos que se persiguen con la creación de esta aplicación,

BowlingStats, la cual es una aplicación que intenta ayudar a los jugadores en su progreso, y a alcanzar el éxito en las competiciones. El objetivo es que se trate de una aplicación sencilla, pero cuya funcionalidad sea mucho más potente que las aplicaciones que en el mercado ya existen.

El capítulo 2 se dividirá en dos partes, la primera de ellas hará una primera aproximación sobre la aplicación a crear, a partir de un pequeño estudio realizado comparando las aplicaciones ya existentes. En la segunda parte del capítulo se hará un estudio del estado del arte, en el que se plantean las principales tecnologías que existen en la actualidad para elaborar la aplicación como se desea en la primera parte del capítulo. De acuerdo con este estudio se llegará a unas conclusiones acerca de las tecnologías a utilizar en el desarrollo de la aplicación.

El tercer capítulo, estará dividido en tres partes. La primera parte consiste en la definición de los requisitos funcionales de la aplicación. La segunda parte definirá la arquitectura que se tendrá en cuenta a la hora de desarrollar la aplicación. Y en la tercera parte se detallará toda la implementación de la aplicación, pasando por los aspectos más importantes de la aplicación. Al final del capítulo, se mostrarán una serie de diagrama de clases, que ayudarán a la comprensión del desarrollo de la aplicación.

En el capítulo cuatro se hará la validación de la aplicación, la cual estará dividida en dos partes. La primera de ellas será una validación funcional en la que se comprobará el correcto funcionamiento de la aplicación. En la segunda parte se hará un estudio con usuarios reales, los cuales harán un test con el que se recogerá la opinión del usuario final para la posible modificación o mejora de la aplicación.

El último capítulo, presentará las conclusiones extraídas de la realización del proyecto y las líneas futuras de este trabajo, con el fin de mejorar el rendimiento de las funcionalidades existentes o crear otras nuevas para que la aplicación sea cada vez más completa.

El final de la memoria incluirá tres anexos. El primero de ellos será la planificación de las tareas necesarias para la realización del trabajo; se incluirá un diagrama de Gant para su comprensión. El segundo anexo contendrá el presupuesto realizado para la elaboración del proyecto, incluyendo todos los gastos necesarios, así como el personal, el material necesario, y el lugar de trabajo. El tercer anexo presentará el marco legal donde se sitúa la aplicación. Para ello, se utilizarán las legislaciones vigentes en el marco español y europeo.

2 ESTADO DEL ARTE

Este capítulo se dividirá en dos partes. En la primera de ellas se explicará la arquitectura de la aplicación. Y a continuación se expondrá un estudio del estado del arte. Se comentarán las diferentes tecnologías que están actualmente en el mercado para desarrollar aplicaciones móviles y, según este estudio, se explicará el porqué de las tecnologías elegidas.

2.1 ESTUDIO DE LAS TECNOLOGÍAS UTILIZADAS

La aplicación BowlingStats constará de dos partes, un sistema operativo móvil sobre el que correr, y además una base de datos en la que ir añadiendo nuestras estadísticas.

Para ello, se ha hecho un estudio previo de las posibles tecnologías actuales en el mercado, comparando entre las posibles tecnologías y argumentando cuál es la mejor para desarrollar la aplicación.

2.1.1 Análisis del mercado actual de sistemas operativos móviles

Actualmente existen tres opciones claramente diferenciadas a la hora de desarrollar para dispositivos móviles:

- Android (*GoogleInk*) (*Michelone 2015)*
- iOS (sistema operativo de Apple) *(RW 2015)*
- Windows Phone (*Windows Phone 2015)*

La cuota de mercado actual se resume en estos datos:

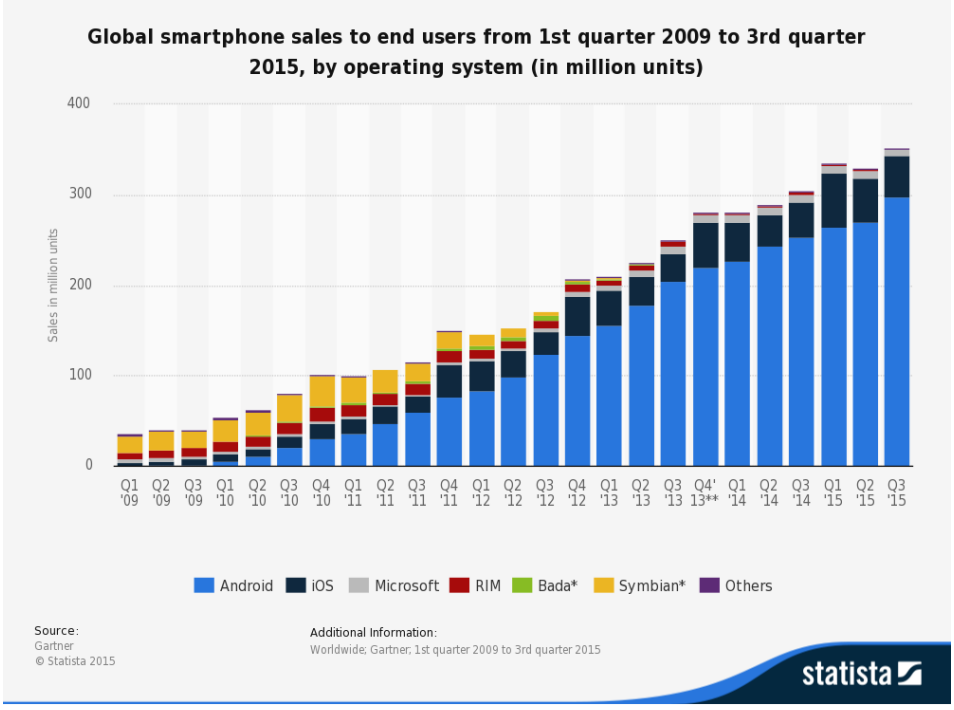

Ilustración 1 Evolución de ventas de teléfonos por Sistema Operativo. (Statista.com 2015)

Hagamos un repaso de la historia de cada una, con sus datos actuales.

2.1.1.1 Android

Aunque ahora mismo el propietario de Android sea Google, en los comienzos fue una empresa externa al gigante de la información. Esta empresa fue fundada en 2003 bajo el nombre de Android Inc., cuyo respaldo económico desde el inicio fue Google. Google la adquirió un par de años después, en el año 2005. Este sistema operativo corre bajo un *kernel* Linux, y la idea original era crear software para los dispositivos móviles que iban en pleno crecimiento de la época en pantallas táctiles.

Este sistema operativo es *Open Source*, a excepción de pequeñas propiedades de la propia marca, Google.

En la comparación con otros sistemas operativos, Android está muy bien diferenciado a la hora de desarrollar el producto final, el dispositivo móvil.

En este escenario actúan tres componentes:

- Google, como desarrollador del sistema operativo;
- los desarrolladores de hardware, empresas que crean dispositivos móviles acordes a las necesidades de este sistema operativo;
- los desarrolladores de software, los cuales crean las aplicaciones para que el usuario disfrute del producto final desarrollado por las dos partes anteriores.

El lenguaje elegido por Android Inc., fue Java para la parte lógica de la aplicación, a la par que para la parte grafica se utiliza XML. Las conjunciones de estas dos tecnologías generan lo que nosotros descargamos del *market*, (APK).

El desarrollo de las aplicaciones es completamente gratuito, dado que se puede desarrollar en cualquier posible PC, con Sistemas Operativos Windows, OS X, Ubuntu… En cualquier computadora se pueden desarrollar aplicaciones Android, por lo que no supone ningún coste adicional. Además, el único coste asociado a las aplicaciones, es a la hora de subirlas al *market*, en el cual sí que habrá que pagar una cuota porque la misma este ahí. Como veremos a continuación en el desglose del resto de sistemas operativos actuales, este coste fijo es único a la hora de subir la aplicación, y no conlleva gastos futuros, lo que lleva a la conclusión de que es más económico.

Además, se incluye la facilidad de puesta en marcha de la aplicación en el mercado, pues no pasa ningún control previo por parte de Android. El control se realiza por tanto a posteriori; para que la aplicación sea retirada del mercado, tienen que producirse acontecimientos que lleven a esa decisión después de haberse producido su subida. Ésta característica, por otro lado, se podría ver como una desventaja, dado que produce que haya muchas aplicaciones no fiables en el mercado.

Desde el inicio de Android en 2007, este ha sufrido una gran evolución pasando por diferentes versiones, las cuales, de manera curiosa siempre fueron llamadas con nombres de dulces por orden alfabético, como se puede ver a continuación:

- Android 1.0 Apple pie
    - Android 1.1 Banana Bread

    - Android 1.5 Cupcake
    - Android 1.6 Donut
- Android 2.0 Eclair.
    - Android2.1
    - Android 2.2 Froyo
    - Android 2.3 Gingerbread. Esta versión fue la más extendida durante un largo periodo de tiempo.
- Android 3.0 Honeycomb- Esta versión duró muy poco en el mercado; estaba más orientada a las tabletas que a los móviles. Pasó muy poco tiempo hasta que se actualizo a la versión 4.x
- Android 4.0 Ice Cream Sandwich
    - Android 4.1,4.2,4.3JellyBean
    - Android 4.4 KitKat
- Android 5.0 Lolipop
- Android 6.0 Marshmallow

2.1.1.2 Windows Phone

Windows Phone es el sistema operativo de Microsoft. Éste surgió como sustituto de Windows Mobile, cuya aceptación, como la de este sistema operativo, nunca ha estado al nivel de Android e iOS.

Este sistema operativo está pensado para un consumo generalista en vez de para un consumo empresarial, como era el caso de Windows Mobile. Ambos sistemas tienen muchas características comunes, sin embargo, Microsoft hizo que las aplicaciones para ambos sistemas operativos no fueran compatibles entre ellos. Como consecuencia se obligó a una entera renovación del sistema operativo forzando a los desarrolladores a crear nuevas aplicaciones.

La primera versión de Windows Phone (7) se ofertó al público en septiembre de 2010. El desglose por las diferentes versiones se muestra a continuación.

- Windows Phone 7 (Photon)
- Windows Phone 7.5 (Mango)
- Windows Phone 7.8: es una versión adaptada, pensada para aquellos dispositivos que no se puedan actualizar a Windows Phone 8.

2.1.1.3 iOS

En enero de 2007 sale a la luz la primera versión del sistema operativo de Apple, acompañado del primer iPhone de la historia. A diferencia de otros sistemas operativos, éste solo se podrá instalar en dispositivos de la propia marca Apple, controlando de esta manera todo el desarrollo del producto, desde el inicio hasta la puesta a la venta.

Además de esto, Apple está presente más allá de la puesta en marcha del terminal. Para poder desarrollar para este sistema operativo, es necesario estar en posesión de un computador de la misma marca. Y a la hora de subir cualquier aplicación, hay que pagar una licencia.

Para más hincapié sobre el control del sistema, Apple verifica la aplicación en cada actualización. De esta manera mantienen en todo momento el control de las aplicaciones del *market* (*AppleStore*). Aunque también es posible desarrollar aplicaciones de manera privada, sin ser necesario subirlas al *market*, éstas tendrán que ser firmadas con alguna licencia, por lo que siempre hay que pagar por tener tu aplicación.

Cada año, es necesario que renovar esta licencia, por lo que el coste de las aplicaciones en esta plataforma es más caro que, en otras plataformas, como, por ejemplo, Android.

Este sistema operativo tiene como lenguaje Objetive-C que es una versión más novedosa del C tradicional. Además, a partir de la versión 6 del programa de desarrollo *xCode* cambiaron el lenguaje de programación al novedoso Swift, que en teoría simplificaría la vida a los programadores, haciendo del lenguaje algo más intuitivo de cara a la programación orientada a objetos.

La evolución de las versiones que ha seguido Apple es la siguiente:

- iPhone OS 1
- iPhone OS 2
- iPhone OS 3
- iOS 4
- iOS 5
- iOS 6
- iOS 7
- iOS 8
- iOS 9

2.1.1.4 Otros sistemas operativos minoritarios

Además de los sistemas operativos mencionados de dispositivos móviles, existen otros de menos importancia, o menos cuota de mercado:

- **Ubuntu**. *(Wikipedia 2015)* Sistema operativo clásico en computadoras, también ha decidido desarrollar un software para dispositivos móviles. Es un sistema operativo basado en Linux desarrollado por Canonical. Este fue presentado el 2 de enero de 2013, la idea principal del proyecto es la de poder utilizar el mismo software tanto en computadoras, dispositivos móviles. La interfaz llamada Unity, se compone, a grandes rasgos, de un menú de aplicaciones a la izquierda, una especie de panel en la parte superior y un sistema de búsqueda que emplea “lentes”.

- **Firefox OS**. *(Corporation 2016)* Como su propio nombre indica, este sistema operativo es el de la compañía del navegador Firefox “*Mozilla Corporation*”, además de la cooperación con otras empresas. Es un sistema operativo para dispositivos móviles, el

cual está basado en HTML5 con núcleo Linux. El sistema operativo está diseñado para permitir que las aplicaciones HTML5 se comuniquen directamente con el hardware del dispositivo bajo JavaScript y Open Web APIs. A finales de 2015 la compañía Mozilla cierra el proyecto, y culmina aquí el desarrollo del sistema operativo.

2.1.2 Elección del sistema operativo

A la hora de la elección del sistema operativo se puede elegir entre tres sistemas operativos ampliamente diferenciados: iOS, Android y Windows Phone. Por cuota de mercado en orden serían: Android, iOS y Windows Phone *(Statista.com 2015).* En cuanto al coste de publicación en el market el orden sería: Android y Windows Phone, iOS. Android y Windows Phone no tienen apenas costes, sólo en la subida de la app al *market*; iOS cobra una cuota anual por cada *app* subida al *market*.

Windows Phone se descarta por su baja cuota de mercado, además del desconocimiento del sistema operativo, y bajo interés en el aprendizaje en esta plataforma debido a la baja cuota de mercado.

IOS se descarta por los costes acarreados en cuanto a su desarrollo se refiere. Para su desarrollo es necesario adquirir un PC de la marca Apple, los cuales son sustancialmente más caros que la media de ordenadores con Windows. Lo que hace interesante el desarrollo en este sistema operativo es el aprendizaje de un sistema operativo nuevo, el cual tiene bastante cuota de mercado actualmente. Sin embargo, se rechaza esta opción por los motivos anteriormente mostrados. *(Yarmosh 2015)*

Por lo que el sistema operativo elegido para el desarrollo es Android, el cual ya ha sido estudiado durante la carrera. Además de no conllevar gastos adicionales, un ordenador de gama media será suficiente para su desarrollo. La subida al market solo conlleva un único pago.

2.1.3 Conclusión

En conclusión, se elige el sistema operativo Android por los conocimientos previos, descartando iOS y Windows Phone pese al interés de los posibles aprendizajes de otras tecnologías actualmente en expansión.

2.2 APLICACIONES SIMILARES EN EL MERCADO

En este apartado se hará una comparación con las aplicaciones que existen actualmente en el mercado. Para ello se mostrará las funcionalidades que ofrecen cada una de ellas, y las ideas que posteriormente han sido usadas en este proyecto.

2.2.1 Bowling Scorer

Bowling Scorer es una aplicación de estadísticas de bolos que ofrece una compleja interfaz para el usuario, mostrando una gran serie de botones y opciones. Como se puede ver en las ilustraciones (*Ilustración 2 Bowling Scorer IIlustración 3 Bowling Scorer II*) se puede observar la complejidad de las interfaces, las cuales presentan un amplio número de funcionalidades de cara al usuario como distintas gráficas de evolución temporal, multitud de estadísticas, etc. Esta aplicación ofrece una versión gratuita limitada y una versión de pago.

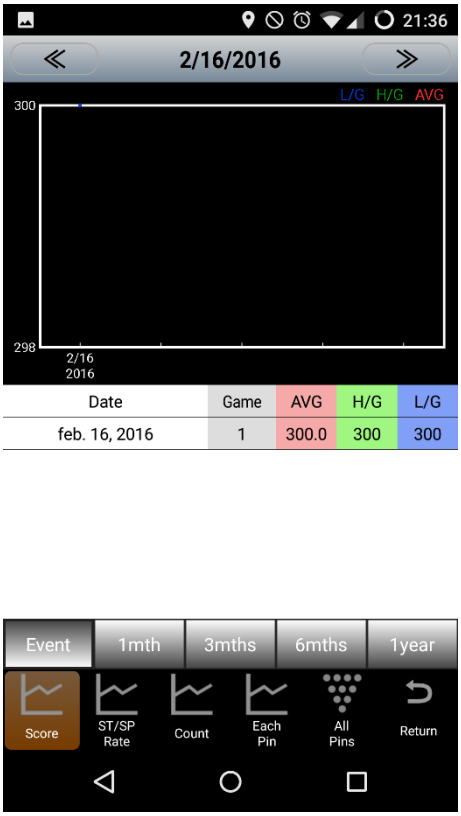

Ilustración 3 Bowling Scorer II

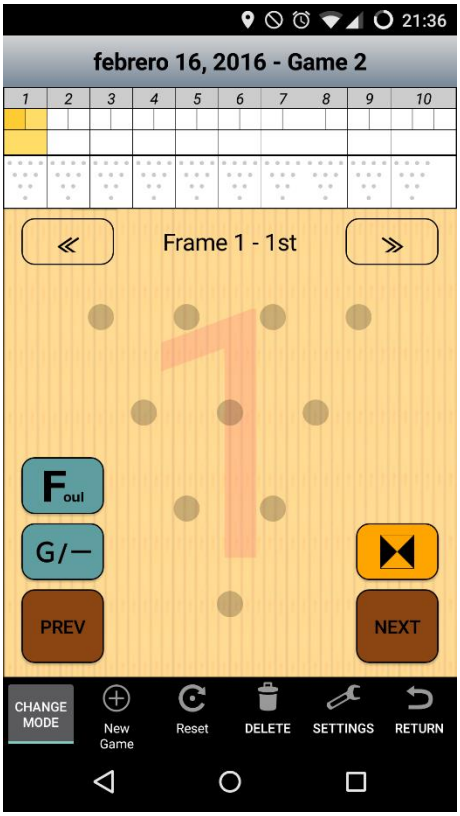

Ilustración 2 Bowling Scorer I

2.2.2 My Bowling

My Bowling es una aplicación mucho más sencilla que Bowling Scorer en la que la interfaz gráfica es bastante pobre, pero que, en cambio, cubre una serie de necesidades básicas para el jugador de bolos. El jugador de bolos podrá insertar sus datos de manera gratuita en la aplicación, y ver sus estadísticas de las partidas mediante un histórico de partidas, como se puede ver en Ilustración 4 My Bowling IIlustración 5 My Bowling II.

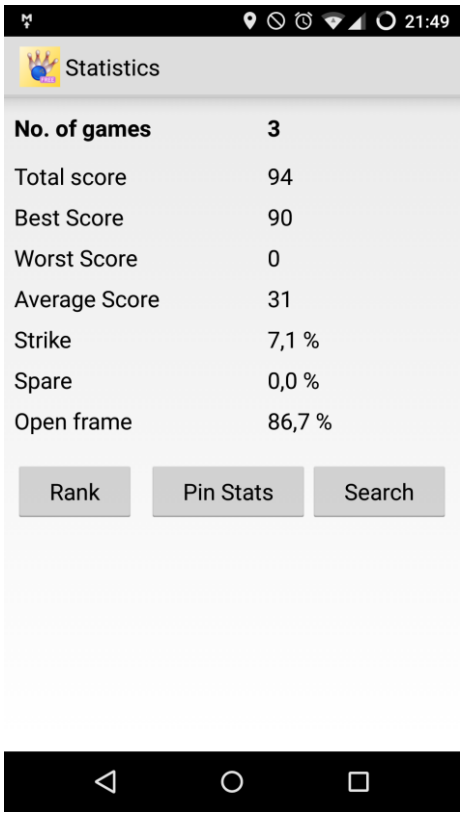

Ilustración 4 My Bowling I

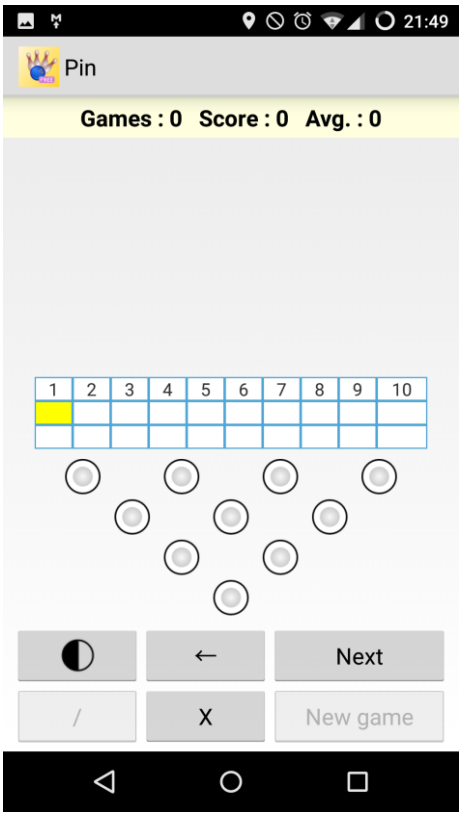

Ilustración 5 My Bowling II

2.2.3 Conclusión

Como conclusión a este capítulo se puede decir que la idea de la aplicación a desarrollar en los siguientes capítulos es algo semejante a estas aplicaciones, pero dando un enfoque más sencillo e intuitivo. La aplicación a desarrollar adoptará funcionalidades tales como: la disposición de estadísticas de combinaciones de bolo, como un histórico de las series de juego. Algunas de las funcionalidades descritas anteriormente, coincidirán con funcionalidades que existen en estas aplicaciones, y se han añadido otras, como la separación del *spare* en combinaciones múltiples y de un solo bolo. Además, la aplicación desarrollada debe ser gratuita.

2.3 FUNCIONAMIENTO DE UNA PARTIDA DE BOLOS

En este apartado se hace una introducción al funcionamiento de la puntuación de una partida de bolos. Para ello, se explicarán todas las posibles combinaciones y el vocabulario estándar de una partida. *(Sportsvite 2016)*

Una partida se compone de 10 tiradas o *frames*. En cada *frame,* se tendrá la posibilidad de haber derribado los 10 bolos o *pins* en una o dos tiradas. A cada tirada se le llamará *shot* a partir de ahora. Dicho esto, por cada *frame* se puede tener estas tres posibilidades:

- *Strike*: se han derribado los diez bolos de una sola tirada, y siempre en el primer *shot.*
- *Spare*: se han derribado los diez bolos en dos tiradas.
- *Open frame:* no se consiguen derribar los diez bolos en dos *shots.*

La puntuación que se va obteniendo durante la partida es simple (*Sportsvite 2016):*

- Strike: al conseguir un *strike,* se puntuará 10 bolos más las dos siguientes tiradas físicas, es decir, si se consiguieran tres *strikes* consecutivos, se tendría 30 bolos en la tirada.
- Spare: al conseguir un *spare*, se puntuará con 10 bolos más la siguiente tirada física, por lo que, si se obtuviera *spare* y en la siguiente tirada se derriban 9 bolos, se puntuaría como 19.
- Open Frame: en este caso se puntuará solamente con los bolos derribados en ese *frame.*

El funcionamiento de la partida se explicará mediante el ejemplo ***(¡Error! No se encuentra el origen de la referencia.)*** de partida con los posibles casos de combinaciones.

AMF FRONTIER LANES

7300 EAST THOMAS RD - SCOTTSDALE

(480) 946.5308

1/30/2008 Score 7:15:20PM

Team Team 3				Lane 3			Game 1				1/30/2008
Player	1	2	3	4	5	6	7	8	9	10	Total
BRIAN	- 8	⑦ -	- 8	7 1	- -	8 -	5 2	- 6	7 -	⑧ -	
Hdcp 0	8	15	23	31	31	39	46	52	59	67	67
JOHN	⑧ 1	6 -	9 -	- -	6 2	1 4	8 1	9 -	4 4	8 / 6	
Hdcp 0	9	15	24	24	32	37	46	55	63	79	79
TOM	X	X	7 /	9 /	9 -	X	7 /	X	9 /	X X 7	
Hdcp 0	27	47	66	85	94	114	134	154	174	201	201

Ilustración 6 Ejemplo de puntuaciones de bowling. (Chapin 2016)

En la segunda tirada, Brian ha hecho dos *open frames*, lo que lleva a una suma de 15, que son los bolos que ha derribado (8 en el *frame* 1 y 7 en el *frame* 2).

En la cuarta tirada, Tom lleva 66 puntos acumulados al finalizar el *frame* 3, y tiene un *spare* en el *frame* 4, consiguiendo en el *frame* 5 *shot* 1 9 bolos, por lo que por la explicación anterior tenemos que Tom consigue una puntuación de 66+10+9=85 bolos en el *frame* 4.

En Tom también podemos observar un doble *strike* en los frames 1 y 2. Esto no suma de manera especial, sino que la consecución de *strikes* seguidos hace que la suma aumente más rápido, dado que un triple, por ejemplo, haría +30 en el que se hubiera iniciado la consecución de *strike* como veremos a continuación.

En el *frame* 1 Tom tiene de puntuación 27 porque tiene X-X-7/, lo que quiere decir que en el primer *frame*, trae acumulado cero, por ser el primer *frame*, más diez del primer *Strike*, más las siguientes dos tiradas físicas o *shots*: X y 7, lo que conlleva a que sume 27 en la primera tirada. Si en cambio hubiera sido X-X-X llevaría 30 en la primera tirada. Y 59 en el segundo *frame*.

A continuación, se definen algunos términos adicionales habituales en las partidas de bolos y que son necesarios para el desarrollo del proyecto.

- *Gutterball: Gutter* significa canal (lateral de la pista), y por tanto, *Gutterball* se produce cuando la bola se cae por uno de los carriles de los laterales, y no se derriba ningún bolo. A esta acción también se le llama canal.
- *Double:* se produce cuando se consiguen dos *strikes* consecutivos.
- Triple: se produce cuando se consiguen tres *strikes* consecutivos.
- *Squad:* serie o conjunto de partidas.

3 DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN

En este capítulo se desarrollará lo relacionado con el diseño e implementación de la aplicación BowlingStats. En la primera parte se abordará lo relacionado a los requisitos funcionales. En la segunda parte del capítulo, se detallarán los detalles de implementación de la aplicación. A partir de diagramas de clases y de flujo, se explicará en detalle la aplicación para su correcto entendimiento.

3.1 REQUISITOS FUNCIONALES

Los puntos de partida para el diseño son los requisitos funcionales de la aplicación, los cuales son:

- la idea original es que se introduzcan las estadísticas de manera sencilla, por lo que la aplicación se desarrollara en una aplicación móvil, la cual es accesible para cualquier persona actualmente;
- la interfaz tendrá que ser sencilla, de cara a que cualquier tipo de jugador pueda, de manera sencilla, insertar las estadísticas de sus partidas;
- el modelo de estadísticas ha de ser escalable para poder realizar el estudio en global, o escalando en series de juego, que es lo que le interesa realmente al jugador;
- La aplicación permitirá un visionado de las diferentes series de juego;
- Las series de juego mostraran un pequeño resumen de estadísticas;
- No habrá en ningún momento ninguna conexión con servidor, todo se hará en local.

3.2 ARQUITECTURA

La arquitectura utilizada en la aplicación será una arquitectura basada en la inserción en base de datos, y la extracción selectiva de los datos para mostrarlos de manera fácil al usuario, para sus posteriores conclusiones.

La base de datos utilizada es la propia base de datos de Android (SQLite), que proporciona una serie de APIs que dan control al programador de manera sencilla. Las tablas de la base de datos serán:

- Frames
- History

Para los datos requeridos necesitaremos estas dos bases de datos de acuerdo a los requerimientos definidos en el apartado anterior. A continuación, se describirán la necesidad y el contenido de estas dos tablas.

Tabla Frames:

Esta tabla se define con el fin de poder guardar en cada partida las posibles combinaciones de bolos derribados, de tal forma que pueda contabilizarse por cada bolo si hubiera quedado de pie, o si hubiera sido derribado.

La decisión de cómo implementar esta base de datos no fue sencilla, pues hubo que sopesar el hecho de que, por cada *frame*, hay otras 10 posibilidades a contemplar, por lo que pudiera haber existido otra tabla del estilo "*Shot*" o tiro, en la cual se contemplarían mediante un identificador, los bolos que hubieran quedado en pie/derribado, esta decisión no fue tomada debido a que se harían demasiadas llamadas a la base de datos, y se pensó que era más eficiente esta otra forma de trabajo

La tabla en sí lo que contiene es:

- identificador del *Squad* o serie a la que pertenece;
- Identificador de la partida, identificador único que hará posible distinguir entre unas partidas u otras;
- Los 10 posibles *frames*.

Una partida tiene 10 *frames,* cada frame tiene dos posibles estados, uno por cada intento de derribar los dos bolos. Esto provoca que, se tengan dos registros en la tabla, uno para el primer intento y otro para el segundo intento. Estos estados, por simplicidad, quedan definidos de la siguiente manera.

Un *String* con 10 valores, ceros y unos, el cero simboliza que el bolo ha sido derribado, y el uno, que el bolo sigue en pie. Lo que quiere decir que un *String* con '0000000000' simboliza el *strike*. De esta manera, tendremos por cada frame, las posibles combinaciones que serán de interés para su análisis estadístico.

En la décima tirada tendremos el caso especial, porque en esta tirada se podría llegar a lanzar hasta 3 veces, por lo que en la base de datos tendremos 3 posibles valores para el *frame* 10.

Tabla History

Esta tabla es creada para la simplicidad de la obtención de los datos a nivel global, sin tener que hacer cálculos dentro de la propia tabla Frames.

Los datos que contiene esta tabla son:

- Identificador de la partida
- Identificador del *squad* o serie
- Fecha de juego
- Bola utilizada
- Lugar de juego
- Bolos totales derribados
- Numero de juegos realizados en esa serie.

Todos estos datos marcarán un resumen de cada serie de juego disputada en un día concreto, por lo que en un mismo día podrían haber sido disputadas diferentes series, lo que es normal en este tipo de competiciones.

3.3 FUNCIONALIDAD

En esta sección se va a hacer un desglose de las distintas funcionalidades de la aplicación BowlingStats

- Comienzo de una serie de juego:
  - El usuario tendrá que rellenar un breve cuestionario, el cual llevará el número de partidas a jugar durante esa serie de juego. Como datos opcionales tendremos la bola a utilizar, y el lugar de juego. Estos últimos datos son opcionales dado que no siempre es de relevancia para el jugador.
  - Después de este breve cuestionario, el usuario empezará a rellenar sus datos de la partida con las siguientes opciones posibles.
    - Al pulsar sobre cada bolo, se marcará como bolo no derribado. Se tomó esta decisión debido a que lo normal en una partida de bolos es que se derriben más bolos de los que quedan en pie, por lo que habría que haber menos pulsaciones de esta manera.
    - Tendrá la posibilidad de marcar una posible bola al canal, no se derriba ningún bolo, de manera sencilla, mediante un botón.
    - También tendrá la posibilidad de volver atrás en caso de poderse haber equivocado.
    - En el caso de no haber hecho pleno en la primera tirada, al realizar la segunda tirada el botón GUTTER, que marcará que la bola se ha caído al canal, cambiará por SPARE, para que el usuario pueda de manera rápida cambiar el estado de todos los bolos a “derribados”
    - En el caso de tener varias partidas en la serie, al terminar la partida, al usuario se le preguntará mediante un pop-up si ha terminado la serie. En el caso de pulsar la opción NO, podría modificar la partida por si hubiera habido algún tipo de confusión
    - De la misma manera, al terminar la serie de partidas, se le preguntará al usuario si ha terminado.
- Pantalla de estadísticas:
  - En esta pantalla se dispondrán de diferentes opciones dependiendo del momento en el que usuario llegue a ella.
    - Si se llega al acabar una partida, tendremos la posibilidad de ver las estadísticas de bolos sueltos, de combinaciones, y un resumen del *squad*, además de un botón para volver a la pantalla de inicio.
    - Si se llega desde la pantalla de inicio, sólo tendremos las posibilidades de visualizar las combinaciones de bolos sueltos, y combinaciones de bolos, habrá un botón

para volver a la pantalla de inicio. Desde esta posibilidad no tiene sentido el resumen, dado que los datos a mostrar son globales.

- La pantalla de estadísticas de bolos será homogénea, tanto para bolos sueltos como para las combinaciones. De esta manera se aprovecha el código para ambos casos, aunque se muestren en pantallas separadas.
  - Esta pantalla estará compuesta por los siguientes elementos:
    - Un elemento grafico donde mostrar qué bolos han quedado en pie, y cuáles han sido derribados;
    - Una etiqueta que muestre el número de ocurrencias, como el número de éxitos.
    - Una etiqueta que muestre el % de acierto con esta combinación.

- La pantalla de resumen nos servirá como detalle de estadísticas de interés en una serie de partidas. Esta pantalla estará compuesta por:
  - Una lista con elementos ordenados en función del número de partidas. Estos elementos serán:
    - Numero de partida
    - Bolos derribados en esa partida
    - Bola utilizada
  - Una lista al pie de la lista anterior con un resumen de las diferentes estadísticas, y cuyos elementos serán:
    - Bolos totales tirados en la serie
    - Tanto por ciento de *Strike*.
    - Tanto por ciento de *Spares* sencillos.
    - Tanto por ciento de *Spares* múltiples.

- Pantalla del histórico. En esta pantalla se mostrará una lista a base de histórico con las diferentes series de juego realizadas durante el uso de la aplicación. Los datos que tendrán que mostrarse serán:
  - La fecha
  - El número de bolos derribados durante esa serie
  - El lugar de juego
  - Resumen de los tantos por cientos

3.4 IMPLEMENTACIÓN

En este apartado se hablará en detalle de la implementación de la aplicación, desarrollando un detallado esquema de las clases utilizadas, el porqué de esta organización, y la necesidad de

las mismas. Se adjuntarán diagramas de clases para la explicación, además de gráficos que ayuden a su comprensión.

En el diagrama presentado a continuación, se puede ver una breve introducción a los posibles flujos que tendrá el usuario a la hora de interactuar con la aplicación.

En primer lugar, se le presentará al usuario una pantalla con tres botones: Jugar, histórico, y estadísticas. El primero de ellos, llevará al usuario a un pequeño cuestionario pre-partida, el cual tendrá que rellenar con al menos el número de partidas a jugar en esa serie. Después de ese cuestionario, pasaríamos a rellenar las estadísticas conseguidas durante el juego, y más tarde, al haber terminado de rellenar todas las partidas, pasaríamos a otra pantalla en la cual tendríamos otros 3 botones: Estadísticas bolos sueltos, estadísticas combinaciones de bolos, resumen. En las dos primeras opciones, como bien se dice en los requisitos, se muestra un gráfico con la combinación dejada, y además datos sobre las veces que hemos conseguido o no realizar ese semi-pleno. La tercera opción, el resumen, mostrara un pequeño resumen de estadísticas sobre la serie.

En segundo lugar, el histórico mostrara, en modo lista, todas las series de juego jugadas durante nuestro uso de la aplicación. Dentro de esta lista, se podrá pulsar sobre cada elemento de la misma, llevando a la pantalla que mostraríamos si acabáramos de haber terminado de jugar esa serie, lo que nos facilitaría de nuevo ver las estadísticas más en profundidad para ese caso en concreto.

En tercer lugar, tendríamos las estadísticas. Estas se refieren a los bolos derribados en todas las posibles ocasiones, es decir, tendremos una estadística global sobre todas las combinaciones, ya sean bolos sueltos, o combinaciones. Se generarán de nuevo dos posibles listas, una para bolos sueltos y otra para combinaciones, separadas en pantallas diferentes.

En *Ilustración 7 Diagrama de Actividades de la aplicación* nos podríamos hacer a la idea como nos podríamos mover entre pantallas:

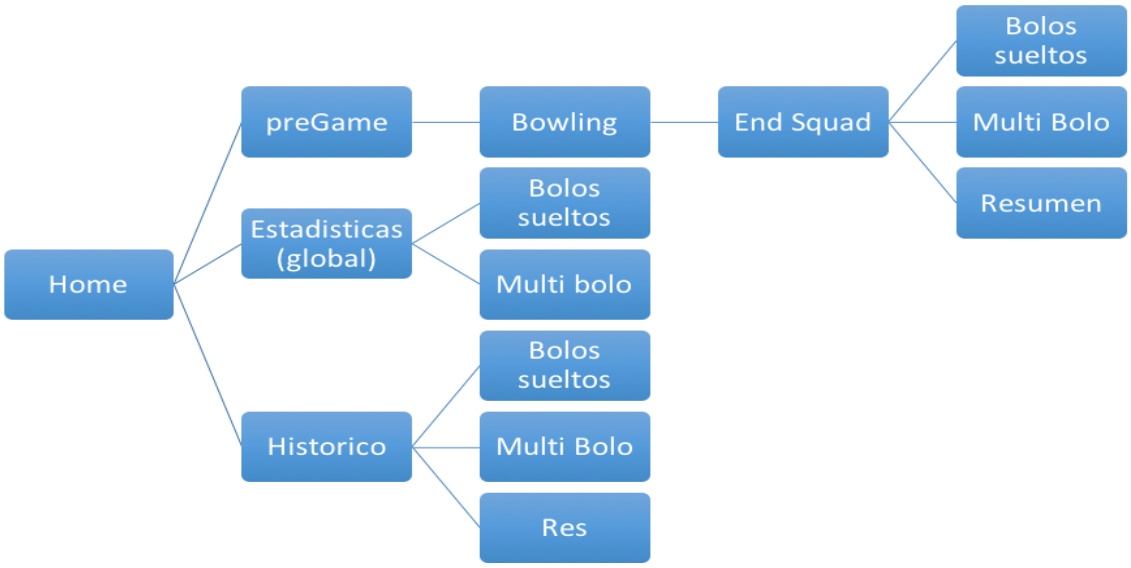

Ilustración 7 Diagrama de Actividades de la aplicación

A continuación, se explicará cómo está desarrolladas cada una de estas pantallas, con sus clases asociadas.

3.4.1 Diagramas de clase

A continuación, se mostrarán varios diagramas de clase, de cara a la buena comprensión de las clases a describir a continuación. Dada la cantidad de clases, se descompondrá en varios diagramas de clases.

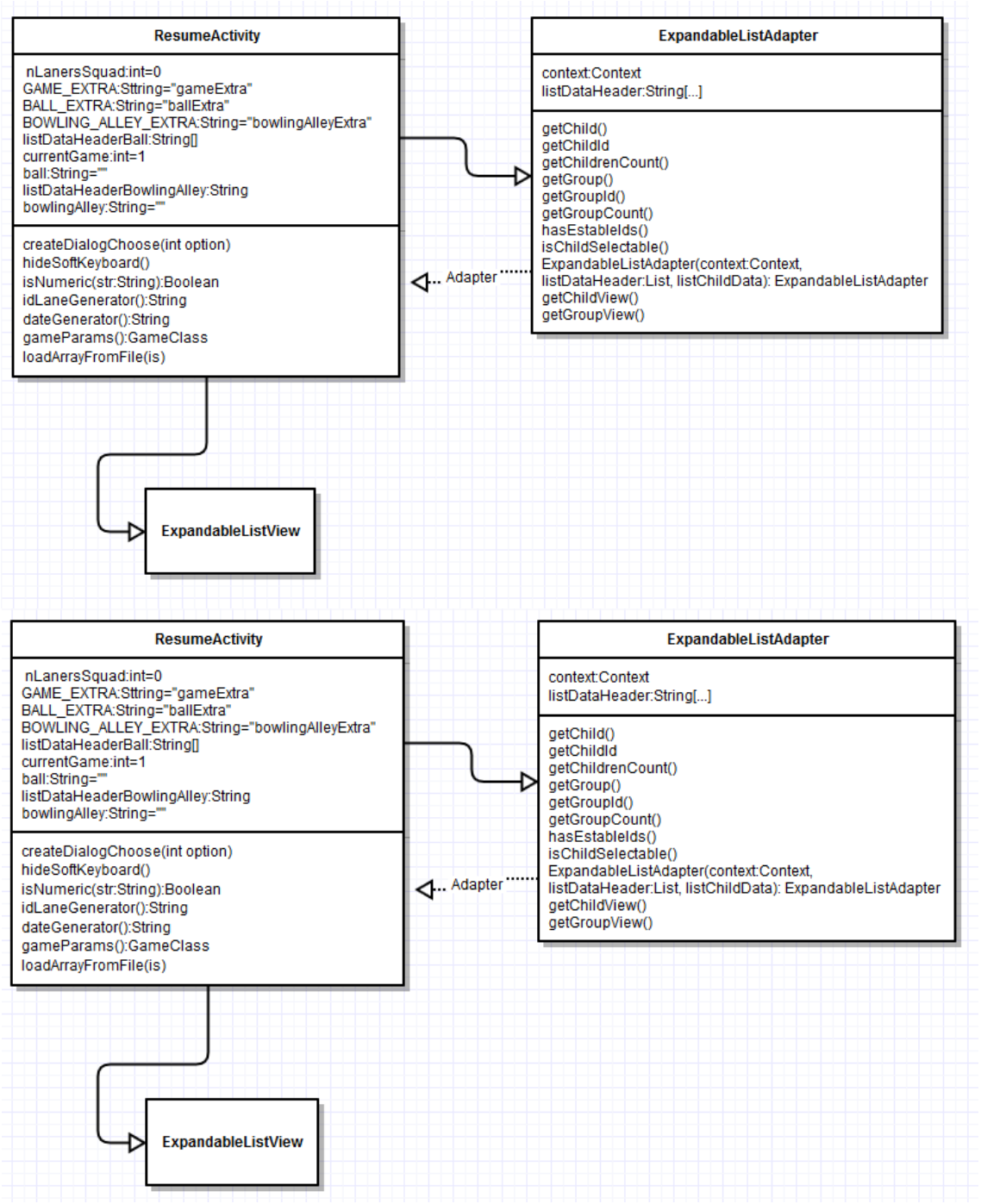

Ilustración 8 Diagrama de clase de la Activity Resume

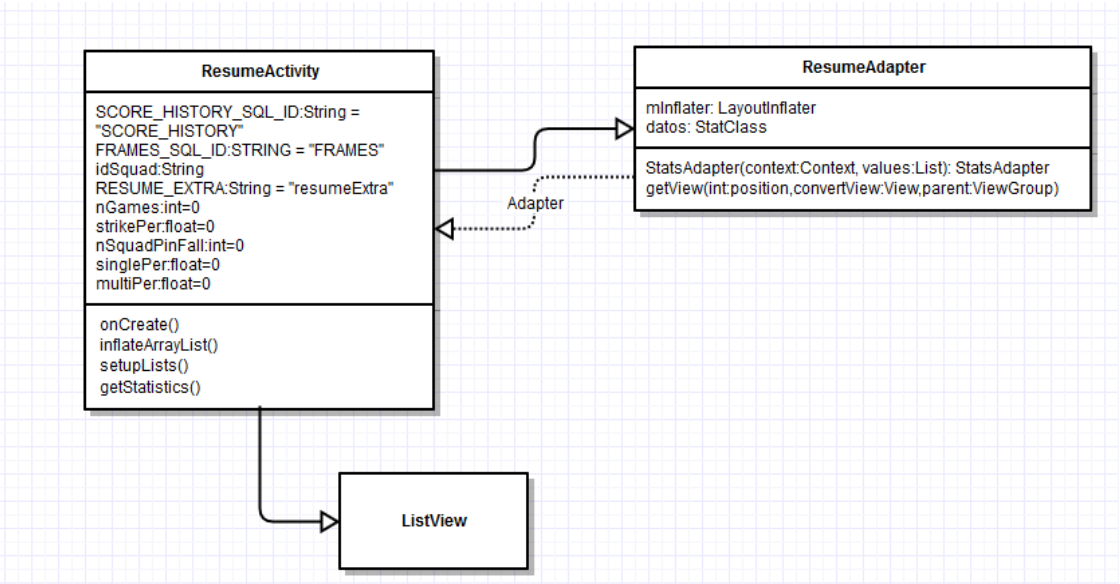

Ilustración 9 Diagrama de clase de la Activity Resume

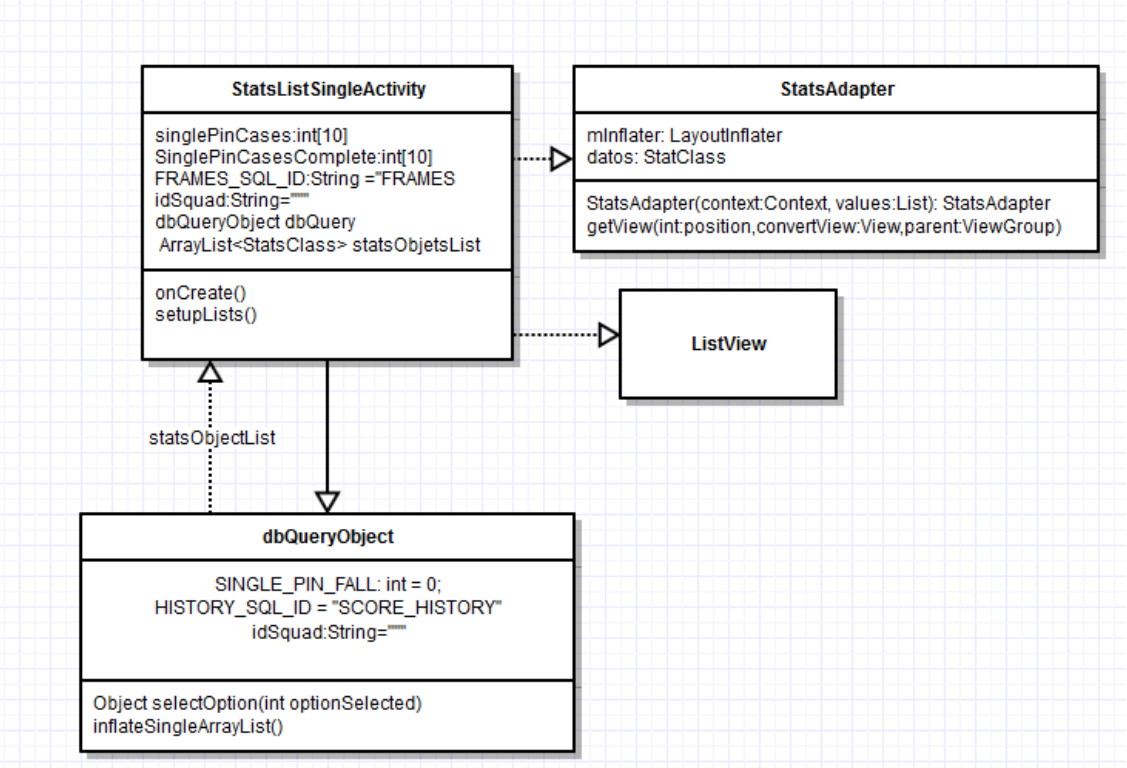

Ilustración 10 Diagrama de clase de la Activity Stats

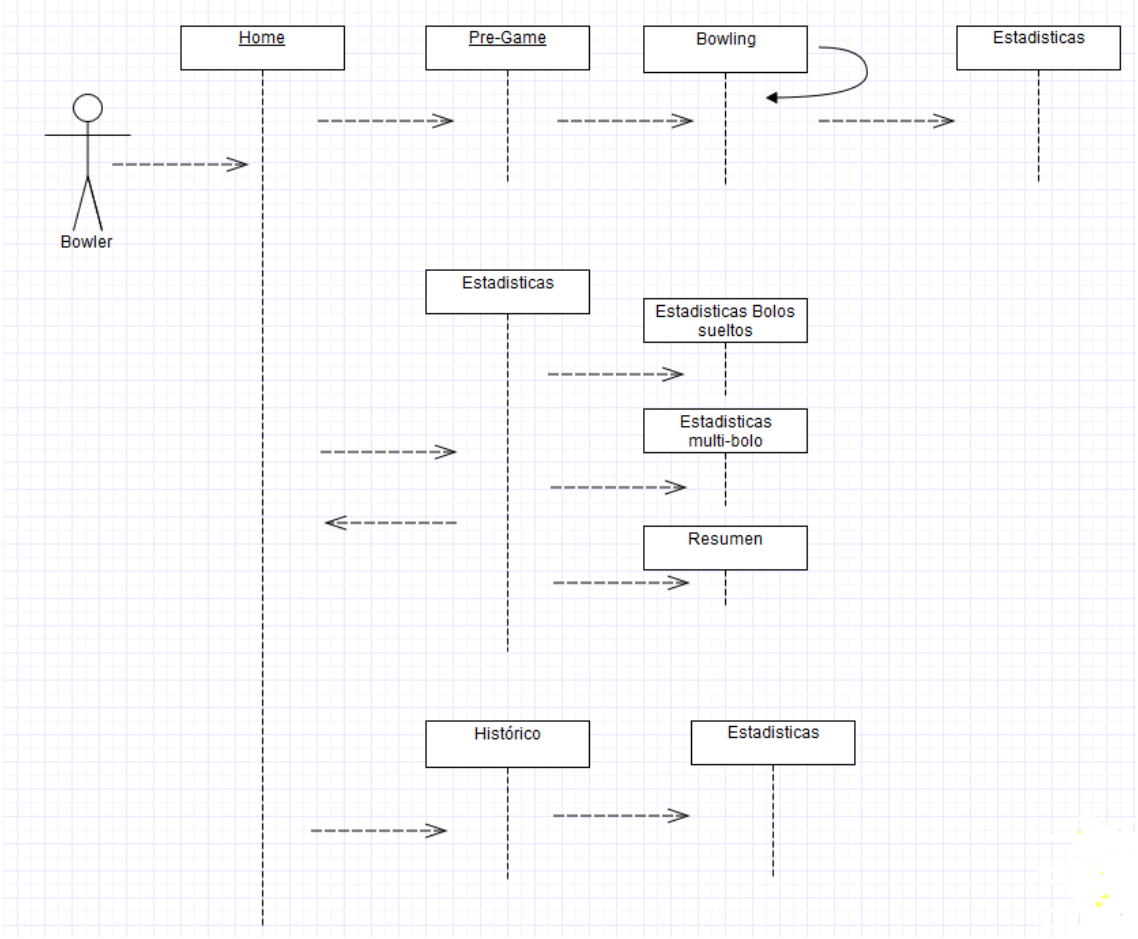

Ilustración 11 Diagrama flujo de la aplicación

3.4.2 Home

Esta pantalla es la principal. A ella llegaremos al acceder a la aplicación, y de ella se distribuye hacia el resto de funcionalidades de la app.

Contiene tres botones, los cuales crean el acceso a:

- Pantalla de pre-game, en la cual haremos un pequeño formulario para pasar a rellenar las partidas.
- Pantalla de elección de estadísticas, que nos llevará a otra pantalla en la que elegiremos si queremos ver las estadísticas de bolos sueltos, o multi-bolos.
- Pantalla del histórico.

Es una pantalla sencilla, sin complejidad computacional que sirve de nexo.

3.4.3 Pre-Game

Esta actividad nos sirve como formulario de cara a que el usuario meta los datos mínimos para poder empezar a jugar una serie de partidas.

Estos datos se incluirán de la siguiente manera:

Para el número de partidas tendremos un cuadro de texto en el que el usuario únicamente podrá escribir números, comprobación clave para evitar errores.

Para la elección de bolera y bola, aparecerá una lista al pie de la actividad. Esta lista será desplegable, es decir, para las boleras, se ha establecido que se organizarán por ciudades, y al expandir una ciudad encontraremos la bolera donde elegir. Y para las bolas estarán organizadas por marcas, dentro de cada marca podremos elegir una bola.

Los datos de las bolas y de las boleras, estarán recogidos en un archivo interno de la aplicación, el cual nos servirá para poder rellenar estas listas. El formato de este archivo es CSV.

Estos datos serán pasados a la próxima actividad, `Activity Bowling`, mediante *extras.* Los datos a pasar son:

- Numero de partidas (obligatorio para poder pasar a la siguiente pantalla)
- Bola
- Bolera

Para poder pasar a la siguiente actividad o pantalla habrá que pulsar el botón situado en la parte inferior "***Go Bowling***".

3.4.4 Bowling

Esta es la actividad más compleja de todas. En ella se irán insertando los datos.

Esta pantalla estará compuesta por:

- Una lista horizontal, que simula el marcador de una bolera, en el cual se irán viendo los resultados obtenidos. Estos se irán actualizando tirada tras tirada.
- Los bolos, situados en las mismas posiciones que en la bolera, de esta manera es muy fácil pulsar sobre los que han quedado en pie.
- Botones:
    - Next
    - Prev
    - Gutter/Spare
    - Botón end (aparecerá solo al finalizar el *squad, y en el caso de acceder desde el historial*)

Esta pantalla sigue unos pasos muy básicos de cara al relleno de los datos durante el juego, los cuales se explican a continuación:

Se presiona sobre los botones, o sobre los bolos caídos tirada tras tirada, cada vez que se quiera avanzar a la siguiente tirada se tendrá que pulsar el botón next.

Cada vez que se pulsa el botón *next*, los datos del marcador se irán actualizando en función de la puntuación obtenida.

Al pulsar sobre el botón *Gutter* (canal) todos los bolos se pondrán en pie, y una vez pulsado el botón *next* se pasaría a la siguiente tirada.

El botón *spare*, el cual solo estará presente si estamos en el segundo lanzamiento de la tirada, hará que los bolos que queden en pie, aparezcan como derribados. Y una vez pulsado el botón *next*, pasaremos a la siguiente tirada.

Una vez terminada la partida, automáticamente aparecerá un cuadro de dialogo informando que hemos terminado la partida. En función de si es la última partida de la serie, o de si hay más partidas posibles por terminar, nos dirá lo siguiente:

- ¿Ha terminado la partida?
- ¿Ha terminado la serie?

Las posibles opciones son las mismas para ambos diálogos, sí o no. Llevando en el primero de los casos a iniciar una nueva partida. Y llevando en el segundo de los casos a la siguiente pantalla.

En esta clase o actividad, se tratan los siguientes aspectos:

Calculo del tanteo de la partida, y la actualización de los datos.

El algoritmo del cálculo de la partida aparentemente es sencillo, pero tiene muchos casos especiales lo que haga que se complique. En el capítulo 2 se tiene una explicación sobre el funcionamiento de una partida de bowling para su correcta compresión.

Tenemos las siguientes posibilidades a la hora de lanzar

- Derribar todos a la primera o *Strike*
- Derribar todos a la segunda o *Spare*
- Dejar el cuadro abierto u *Open Frame*.

Estas son las tres posibles combinaciones posibles en un cuadro o tirada. Las tiradas se descomponen en lanzamientos físicos, su argot en inglés, que es de más fácil comprensión es:

- Tirada-*Frame*
- Lanzamiento-*Shot*

A partir de ahora se tratará con la traducción en ingles ya que es de más facilidad comprensiva.

Como se puede ver *en Ilustración 12 Ejemplo de hoja de anotación de puntuaciones de bowling.*, los colores verdes corresponden a lo que denominamos los *frames*, y los colores marrones, a lo que denominamos *shots*. Los *shots* son en realidad cada tirada física realizada, es decir, un *Strike* es la consecuencia de derribar 10 bolos a la vez en un *shot*, y un *spare*, es la consecuencia de haber derribado los 10 bolos, pero en este caso, en dos *shots*

.

Ilustración 12 Ejemplo de hoja de anotación de puntuaciones de bowling.

Una vez explicado esto procedemos a explicar cómo se realiza el tanteo de cada tiro.

Un *Open Frame* vale tanto como los bolos que se hayan derribado. Puede ir desde 0, si no se ha derribado ningún bolo en los dos intentos posibles, hasta 9.

Un *Spare* vale 10 bolos más en el siguiente *shot*. Es decir, en el caso de haber hecho *Spare* y en el siguiente *shot* 8, puntuaría como 18, sumado a la puntuación obtenida en los lanzamientos anteriores. Si hacemos *Spare* y en el siguiente *shot Strike*, puntuaría como 20.

Un *Strike* vale 10 bolos más los siguientes dos *shots*, es decir, se sumarán diez bolos más las dos tiradas físicas que realice el jugador, siendo independiente de la puntuación que consiga en ambas tiradas físicas, siendo posible que fuera, tanto dos *strikes,* como un *spare*, como un *open frame.*

Además, añadir que, en la última tirada, se pueden realizar hasta 3 *shots*, solo en los casos que hayamos conseguido, *Strike* o *Spare*. Como veremos a continuación.

Ahora se explicará uno de los algoritmos con más casuística de toda la aplicación.

Es el método `scoreAcumulated` el que va calculando por cada *frame*, la puntuación obtenida. Se le llamará cada vez que se tenga que actualizar el marcador, por lo que cada vez que pasemos al siguiente *frame*, se llamará a este método.

Este método depende de diferentes casuísticas. En función de en qué *frame* estemos situados, las posibilidades son las siguientes:

- *Frames* #1-8 Estos frames son tratados por igual dado que se comportan de la misma manera. Todos son iguales entre sí. La diferencia con el resto se explicará más adelante. Estos *frames* tienen las posibles posibilidades de ser
  - Un *Open Frame*, por lo que se sumaría al acumulado lo que tengamos en este *frame.*
  - *Spare*, por lo que tendríamos que esperar a tener el siguiente *shot*, y hasta este no actualizaríamos la vista
  - *Strike*, necesitaríamos los dos siguientes *shots*, por lo que con la misma intención del *Spare*, tendremos que esperar a tener esa información. En el momento en el que la tengamos se actualizará con su valor correspondiente

- *Frame* 9, este frame es especial porque es precedente al *Frame* 10, el cual es especial por poder tener hasta 3 *shots*. Esto conlleva a que el número 9 no pueda ser tratado como los demás. El noveno frame puede tener la posibilidad de un Strike, y que el primer shot de la tirada 10 fuera también un strike, es decir, tendríamos un doble, si lo tratáramos de la misma manera, nos saldríamos del array, yendo a buscar en la posición 11, la cual no existe para este problema en cuestión.

  Para solucionar este problema, tendremos que esperar hasta tener los datos necesarios, haciendo la comprobación de los mismos en el momento en el que estén accesibles, en el momento de que se disponga de los datos, se hará uso de ellos calculando el valor del *frame*.

- *Frame* 10. Este *frame* se trata de manera diferente, ya que se suman los valores íntegros de lo que hayamos conseguido. Es decir, en el caso de conseguir un triple en la tirada 10 (XXX) sumariamos 30 en la misma tirada, y en el caso de conseguir por ejemplo un *open frame* de valor 7, se sumaría al acumulado de la misma manera que

el triple. Por ello este *frame* se trata de manera diferente al resto. No es necesario esperar a ningún *frame* posterior para poder realizar su cálculo.

Otro método de interés en esta clase es el método `drawScore` que es el que se encarga de pintar en el score los valores conseguidos. Al pulsar sobre *Next*, se llama a este método.

Lo primero que hace este método es saber cuántos bolos hay en pie en ese momento, mediante un String que nos marca que bolos están en pie y cuáles no, esta forma es la utilizada también para insertar en base de datos *frame* a *frame*, la cual ya fue explicada anteriormente.

Una vez que sabemos qué bolos están en pie/derribados hacemos una comprobación que estamos en el rango de valores estipulado, y no nos hemos salido del array.

En este método solo se hace de manera especial el ultimo *frame*, dado que es el único que se puede pintar de manera diferente, pudiéndose pintar hasta tres *shots* en el.

Se harán comprobaciones para saber dónde estamos mediante 2 variables de estado, que marcarán si estamos en el *shot* uno del tiro, o el dos, o el tres en el último *frame*. Y además tendremos otro estado para saber en qué *frame* estamos.

Cada vez que pasemos a actualizar el estado del número de *frame*, es decir sumaremos uno para pasar al siguiente, además se volver a llamar a scoreAcumulated, se añadirá a la base de datos ese *frame*, y se cambiará el estado para que vuelva a estar en el primer *shot*.

En el momento que sumemos uno para pasar al siguiente frame, se invocará al método scoreAcumulated. Al mismo tiempo, se añadirá a la base de datos ese *frame*, y el estado del *frame* volverá a estar orientado al primer *shot.*

En esta clase además hay otros métodos auxiliares que nos ayudarán a crear los Strings en función de los bolos caídos, a insertar en base de datos, a saber, si el próximo *shot* está disponible para poder actualizar el resultado.

La inserción en base de datos, como ya vimos en la arquitectura, se hará mediante una petición con el identificador de la partida, y además un segundo identificador que será el del *squad*, o conjunto de partidas. En este método se tienen en cuenta las diferentes posibilidades:

- Que sea el primer *frame*, por lo que hay que generar esa línea, y meter los primeros valores de la tabla.
- Que sea el último *frame*, que será actualización de la línea, el cual será diferente pues tiene un *shot* más posible, por lo que hay que tratarlo de diferente manera.
- El resto de *frames* serán tratados como actualizaciones de esta línea, actualizando los valores necesarios de su respectivo *frame*.

También tendremos en esta clase los creadores de los identificadores únicos, tanto para el *squad*, como para la partida, ambos identificadores se generarán con el generador de códigos que proporciona Android, añadiendo una breve etiqueta para diferenciar los *squads* de los *games*.

3.4.5 Activity Estadísticas bolos

Esta clase englobará las dos actividades referentes a mostrar las estadísticas de las combinaciones de bolos. Como bien se explica en los *Requisitos funcionales* de la aplicación, se tendrá:

- Una parte visual donde se podrán ver los bolos en pie y derribados (la combinación)
- Una etiqueta que nos dirá las veces que nos ha ocurrido esta combinación, y las veces que la hemos conseguido realizar con éxito.
- Una etiqueta que nos marque el % de las veces conseguidas.

Cada combinación se marcará como un elemento de la lista. Para realizar esta lista tendremos varias cosas: cómo generar la lista a través de llamadas a la base de datos, el adaptador encargado de llenar cada elemento de la lista, y en el caso de las posibles combinaciones múltiples de bolos, el filtro para coger únicamente las combinaciones por bolos en pie. Para ello, explicaremos paso por paso.

En primer lugar, tendremos que extraer de la base de datos las posibles combinaciones. Distinguiremos dos casos, el caso de los bolos sueltos, el cual es el más sencillo, y el caso de los casos multi-bolo. Explicaremos primero el segundo caso, dado que es más complejo, y una vez explicado volveremos al primero, el cual, una vez tengamos la lista de posibles casos, se hará de la misma manera.

Para conseguir las posibles combinaciones se hará una llamada a la base de datos de la siguiente manera, se creará un objeto de la clase `dBQueryObject` el cual permitirá hacer llamadas a la base de datos con una opción de entrada. Para este caso se utilizará la constante "*MULTIPLE_PIN_FALL"* que nos devolverá una lista de Strings con las distintas ocurrencias en la tabla. Para ello, dado que no se consiguió una *query* que devolviese justo lo que se necesitaba, se procedió a la siguiente solución:

- En primer lugar, cojo todas las posibles combinaciones.
- Después se va haciendo *matching* una a una, y se copia a otra lista que estarán las distintas posibles combinaciones.
- Al tener ya todas las posibles combinaciones, nos disponemos a contar cada una de ellas, para ello hacemos otra llamada a la base de datos: "*SELECT COUNT (*) FROM FRAMES WHERE FRAME_X_X = combinación"*, vamos recorriendo los diez *frames*, y sumando lo obtenido, de esta manera obtenemos el resultado de las posibles combinaciones.
- A continuación, se devuelve una lista de objetos de la clase `StatsClass` la cual tendrá diferentes datos de interés.

Una vez tenemos la lista de la clase StatsClass, pasaremos a crear el adaptador, que es el que se encargará de pintar cada elemento de la lista, pasándole una lista de elementos.

Una vez creada la pantalla, en el caso de los bolos sueltos ya estaría, pero en el caso de los multi-bolos, tendremos la opción de filtrar por número de bolos en pie, es decir, tendremos las opciones de todas las combinaciones, y, además, podremos filtrar pos las veces que nos hayan quedado X bolos en pie.

Para ello, lo que se hace es pasar a una lista auxiliar las combinaciones que coincidan con nuestra opción, y volver a refrescar el adaptador, y por consiguiente la lista.

3.4.6 Activity Resumen

Esta actividad será la encargada de mostrar un resumen de las estadísticas de la serie. Los datos a mostrar serán:

- En una lista, las partidas jugadas:
    - Numero de partida
    - Bolos derribados en esa partida
    - Bola utilizada
- Al pie de esta lista, tendremos otra serie de datos relevantes para la serie:
    - Numero de partidas
    - Total de bolos derribados en esa serie
    - % de *Strike* o pleno
    - % de *Spare* o semipleno de bolos sueltos
    - % de *Spare* o semipleno de combinaciones multi-bolo

La lista se formará de la misma manera que en el caso de las estadísticas, siendo esta vez el objeto "`ObjetoResume`" el cual solo tiene 3 variables: número de partida, bolos, y bola. Para ello se utiliza también otro archivo de la capa visual, que a la vez tendrá 3 etiquetas que son los que conformarán cada uno de los elementos de la lista.

Para las estadísticas en el pie de la actividad, lo que tendremos son diferentes llamadas a la base de datos, para obtener los mismos. Estas se harán a través de la clase `dbQueryObject`. Los datos necesarios para hacer los cálculos de las estadísticas serán:

- Número de *strikes* realizados
- Número de ocurrencias/éxitos en los bolos sueltos.
- Número de ocurrencias/éxitos en los multi-bolo.
- Suma de los bolos totales derribados

3.4.7 Activity Histórico

Esta será la encargada de mostrar una lista con todas las series jugadas durante la historia del usuario. Cada elemento de la lista, el cual tiene un objeto asociado en su creación de la clase `ObjetoHistory`, contendrá la información de cada etiqueta de la lista.

Como en cada caso de las vistas con listas, tendremos un *adapter* asociado a esta clase, que será el que nos rellene la capa visual, a través de un elemento *layout*, que será el que nos aporte la capa visual.

Lo que se mostrará en cada elemento de la lista es:

- Fecha
- Número de partidas
- Bolos derribados en esa serie
- % de *Strike*
- % de *Spare*, tanto múltiples como bolos sueltos.

Para ello, de la misma forma que en la clase anterior, utilizaremos un objeto de la clase *dbQueryObject*, para hacer las llamadas a la base de datos y obtener los resultados. Para ello utilizaremos el id de la serie para identificar qué casos son de interés.

Una vez pulsado en cualquier elemento de la lista, nos llevará a la pantalla de las estadísticas mostrada nada más terminar la serie, la cual tendrá los 3 botones: Est. Bolos sueltos, Est. Multi-bolo, resumen. Para poder visualizar de nuevo y, en más profundidad estas estadísticas ya pasadas.

Ahora se realizará una explicación de las clases que anteriormente se han comentado, para su correcto entendimiento. Algunas de ellas serán englobadas dentro de una superclase, debido a que el funcionamiento de las mismas es el mismo, y su interacción también, solo se mostrará la relación entre ellas.

3.4.8 Objeto-Lista(objeto)-Adapter-XML asociado.

Aquí se explicará cómo se rellena cada objeto de una lista, dado que se tienen varios ejemplos para explicar. Quedará resumido aquí como se tratan este tipo de listas, y de esta manera, englobará todo el proceso en una sola explicación. Más adelante se mostrará un gráfico con las clases asociadas a cada elemento de cada lista de la aplicación.

A la hora de crear una lista, tendremos que hacer uso de diferentes elementos, los cuales son:

- Objeto a utilizar
- Lista de objetos, que a la par serán los datos a mostrar en la lista
- *Adapter*, o clase que se encarga de rellenar los objetos en la capa visual de la aplicación
- XML, o capa visual de la aplicación, que contendrá los elementos gráficos.

Para ello, el procedimiento es el siguiente

1. Crear la lista, o rellenarla con los objetos deseados.
2. Una vez tenemos la lista creada, crearemos un objeto de la clase *Adapter*, pasando en el constructor la lista de objetos que deseamos crear.
3. Al objeto visual asociado a nuestra lista, la asignamos ese *adapter*

Estos son los pasos que se han seguido a la hora de crear las listas. En *Ilustración 13 Relación entre clases y XML* se pasará a mostrar cómo se relacionan las diferentes clases y elementos visuales (XML) en esta aplicación.

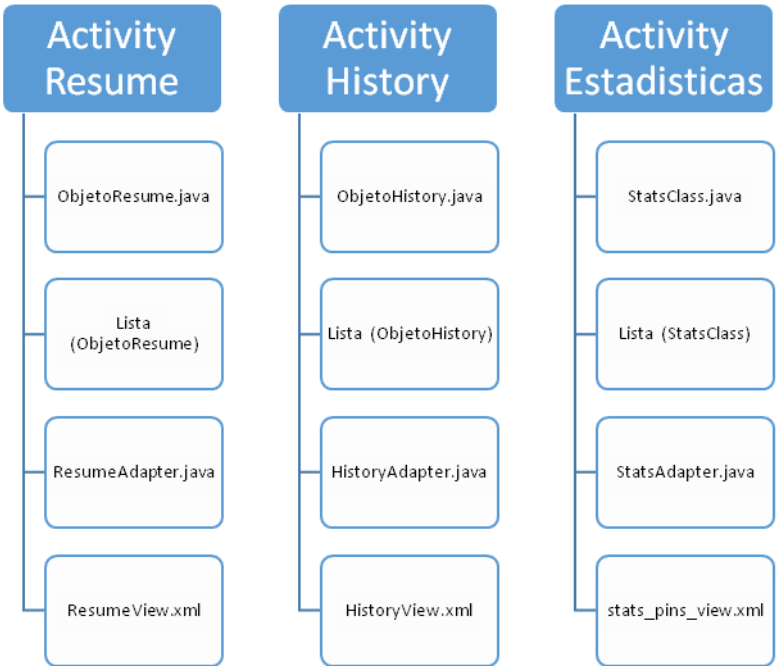

Ilustración 13 Relación entre clases y XML

3.4.9 dbQueryObject

Esta es la clase que utilizaremos para hacer las consultas a la base de datos, para ello, al crear el objeto le pasaremos una opción dada, llamando al método `selectOption`, el cual devolverá un objeto genérico, y tendrá de dato de entrada la opción requerida. Devuelve un objeto genérico, debido a que éste método puede devolver tanto listas como números, dependiendo de la opción que le requiramos. Las opciones posibles son:

- ***Int SINGLE_PIN_FALL = 0.*** Opción que devuelve una lista con las distintas combinaciones de bolos sueltos.
- ***Int SINGLE_PIN_ CONT = 1.*** Opción que devuelve tanto el número de ocurrencias, como el número de éxitos de los bolos sueltos.
- ***Int MULTIPLE_PIN_FALL = 2.*** Opción que devuelve una lista con las distintas combinaciones multi-bolo.
- ***Int MULTIPLE_PIN_ CONT = 3.*** Opción que devuelve tanto el número de ocurrencias, como el número de éxitos de los bolos sueltos.
- ***Int STRIKE_CONT = 4.*** Opción que devuelve el número de *strikes* conseguidos.

Gracias a esta clase, se obtienen estos datos, y a partir de ellos se extraen diferentes estadísticas, por ejemplo, para las tantos por ciento, se hará un cálculo sencillo en las diferentes clases, para hallar ese número.

Para resumir, dado que se hace de la misma manera para los bolos sueltos, como para los múltiples, lo que se hará es englobarlo de la misma forma.

Para extraer esta información de la base de datos, dado que se tienen 2columnas, 10 de primer *shot*, 10 de segundo *shot*, +1 *shot* del ultimo *frame*. Se tiene que ir recorriendo cada columna para ver si esta nuestra combinación, la *query* utilizada es:

```
dbQueryFShot = "SELECT COUNT(*) FROM " + FRAMES_SQL_ID +
                    " WHERE " + shots[0] + " = '" + pinsDown + "'" +
                    "AND ID_SQUAD = '" + idSquad + "';";
dbQuerySShot = "SELECT COUNT(*) FROM " + FRAMES_SQL_ID +
                    " WHERE " + shots[0] + " = '" + pinsDown + "' " +
                    " AND " + shots[1] + " = '" + "0000000000" + "'" +
                    "AND ID_SQUAD = '" + idSquad + "';";
```

De esta manera se va seleccionando, y guardando en variables estos datos. Como se puede observar, para el segundo *shot*, se hará la comprobación si teniendo el primero *shot*, y el segundo se produjo el *spare*, para así tener el número de éxitos.

Para seleccionar el número de Strikes, lo que se hará es una *query* parecida a la anterior, recorriendo las distintas columnas de la base de datos, buscando solo en los primeros tiros cuando se produjo el *Strike*. De esta manera se hará el conteo de los *Strikes* obtenidos, a la vez que se hallará el número total de lanzamientos para poder obtener el % de *Strike*.

3.4.10 Objetos de clase

Estos objetos son los que se utilizarán para guardar información para rellenar las listas. Son clases de tipo *getter* and *setter* y que apenas hacen un tratamiento de datos en su interior.

3.4.10.1 BowlingBall

Tiene dos variables en su interior, marca y modelo. Se utiliza al generar la lista de selección de bola. Es útil para el adaptador capaz de generar una lista visual a partir de la lista de objetos de esta clase.

3.4.10.2 FrameClass

Tiene tres variables de tipo *String* que componen la clase. Estos *Strings* hacen alusión a los diferentes *shots* posibles en los frames:

- Primer *shot*
- Segundo *shot*
- Tercer *shot*

Además de los métodos de acceso, y de sus constructores, en esta clase sí hacemos uso de las variables.

Tenemos métodos que nos proporcionarán resultados en función del momento de la partida en el que estemos. Por ejemplo, si venimos de un doble (XX), nos hará el cálculo de cuantos bolos estaríamos acumulando en ese frame.

También habrá métodos de comprobación, como `parseShots`, que nos permitirá obtener un número en función, del String que le pasemos; por ejemplo “X” y “/” serían diez bolos. Esto será de utilidad a la hora de hacer el recuento y los cálculos.

En resumen, en esta clase se ayuda a la realización cálculos, para que queden más simplificados y de manera más sencilla, de cara al código y su comprensión.

3.4.10.3 GameClass

Los objetos de esta clase serán pasados entre actividades para poder obtener la información necesaria a la hora de ser mostrada, o de buscar en base de datos. Las variables de esta clase son:

- `String idSquad`, el identificador del *squad o* serie de partidas.
- `String idLane`, el identificador de la partida actual.
- `Int currentLane`, número de la partida en juego.
- `Int totalLanes`, número de partidas en total a disputar.
- `String date`, fecha de realización de las partidas.
- `String ballUsed`, bola utilizada durante el *squad*.
- `String bowlingAlley`, lugar de juego.

Esta clase no implementará nada adicional. Sólo servirá para simplificar código. Sólo contendrá métodos de acceso a los datos.

3.4.10.4 ObjetoHistory

Es el objeto utilizado para llenar la lista del histórico. Los datos que contiene son:

- `String date`, que contendrá la fecha de cada squad.
- `String nGames`, que contendrá en formato texto, el número de partidas jugadas.
- `String pinFall`, que contendrá el número de bolos derribados en formato texto.
- `String strikePer`, tanto por ciento de *strikes*.
- `String sparePer`, tanto por ciento de *spares*.
- `String idSquad`, el identificador del *squad*, que no mostraremos pero que sí nos será necesario si el usuario pulsa y hay que pasar a la siguiente pantalla; nos permitirá seleccionar los datos de la DB.

No hay métodos más haya que los de acceso a los datos.

3.4.10.5 ObjetoResume

Como en la clase anterior explicada, esta clase solo sirve para rellenar una lista. Los datos que contiene son:

- `String Ball`, bola utilizada durante el *squad*.
- `String Score`, puntuación obtenida en la partida.
- `String nGame`, número de partida en la que estemos.

Tampoco hay métodos extra de tratamiento de los datos, solo métodos que permiten el acceso y uso de las variables propias de la clase.

3.4.10.6 StatsClass

Clase que nos ayuda a tratar los datos de las estadísticas. Esta clase sí contendrá un método para tratar los datos de manera sencilla, que veremos ahora.

- `String pinState`. Es un String de la combinación dejada. Nos ayudará a pintar los bolos ya estén derribados o no. El formato de este String será de diez caracteres, de la forma"0011001100", de manera que se tienen en cuenta que el orden de los caracteres está asociado a los bolos. El primer carácter corresponderá con el primer bolo y así sucesivamente.

- `Double knockedDown`. Tendremos las veces que conseguimos derribar la combinación.

- `Double total`. Representará las ocurrencias totales de la combinación en cuestión.

- `Double percentage`. Es un número que corresponde con el porcentaje de las dos variables anteriores: (knockedDown/total) *100.

Se utilizan métodos de acceso a las variables, y además tendremos un método sencillo. Este método permite hacer el cálculo del porcentaje de manera automática. Dividiendo la variable `knockedDown` entre el total, y el resultado de este lo multiplicamos por 100.

4 VALIDACIÓN

En este capítulo se expondrán las validaciones realizadas para que la aplicación cumpla con los requisitos funcionales expuestos con anterioridad. Para este propósito, se realizarán diferentes pruebas para poder demostrar que la aplicación cumple con dichos requisitos.

4.1 VALIDACIÓN DE REQUISITOS FUNCIONALES

Según lo expuesto en el capítulo 3, la aplicación tiene 2 áreas bien diferenciadas, la primera de ellas es la inserción de los datos en la aplicación, y la segunda, la visualización de los mismos mediante listas. Estas listas abarcan tanto las estadísticas de las combinaciones conseguidas, como las series de juego jugadas, como un breve resumen de los datos de una serie de juego. Por este motivo hay que hacer una serie de comprobaciones de que todos esos datos sean correctos. Para ello se harán diferentes pruebas, tanto de inserción de los datos, como de la visualización de los mismos.

4.1.1 Inserción de datos

Durante esta prueba comprobaremos que, al pulsar sobre los bolos, e ir dando al botón NEXT, se actualiza el marcador, y a la vez se va haciendo el correcto cálculo del marcador en cada *frame*.

Para ello, mostraremos una serie de capturas de pantalla, que conformarán un flujo para su demostración. En esta primera prueba realizaremos los siguientes pasos:

Una vez en la pantalla principal, pulsamos sobre "*next*" y como está diseñada la aplicación, eso sería un *strike.*

En la siguiente tirada pulsaremos sobre dos bolos, y pulsamos "*next*". Al estar en la segunda tirada, pulsamos sólo sobre uno de ellos. Pulsamos "*next*".

En la siguiente tirada pulsamos "*next*" para marcar un *strike*.

Como podemos observar en *Ilustración 14 Inserción de datos*, el funcionamiento es correcto de acuerdo con los requisitos funcionales de la aplicación, actualizando correctamente el marcador, y cogiendo bien los bolos derribados.

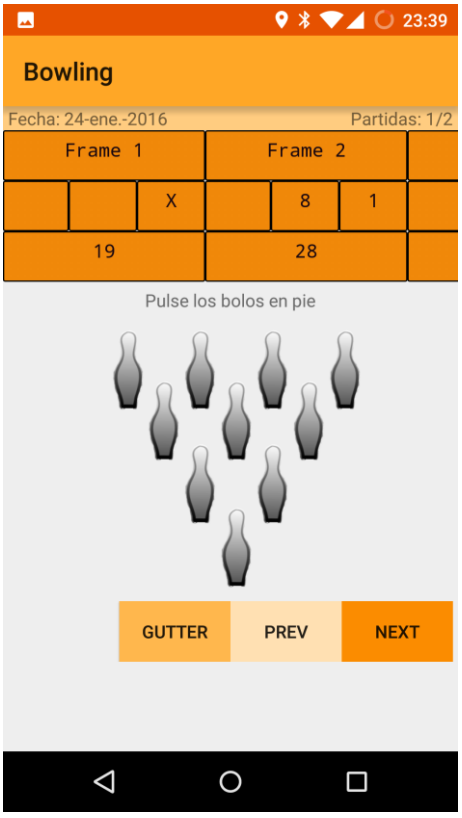

Ilustración 14 Inserción de datos

4.1.2 Extracción de estadísticas

En esta validación comprobaremos que extraemos correctamente de la base de datos los resultados obtenidos durante la serie, para así mostrar las estadísticas. Para ello, habremos introducido los datos de la figura *Ilustración 15 Partida ejemplo (Tom) (Chapin 2016):*

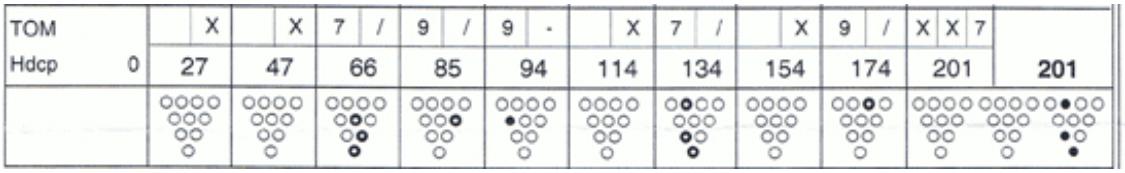

Ilustración 15 Partida ejemplo (Tom) (Chapin 2016)

Como se puede observar en esta partida tendremos que Tom tiene toda clase de atributos: *strike, spare, open frame,* lo que nos facilitará la comprobación de distintas estadísticas.

La primera comprobación a realizar es los bolos sueltos: para ello pulsamos en "IR A ESTADÍSTICAS DE BOLOS SUELTOS"

Como se puede observar en las imágenes, tenemos que en *Ilustración 16 Estadísticas bolos sueltos I* y *Ilustración 17 Estadísticas bolos sueltos II* se puede confirmar que el bolo 4 no se ha derribado en el segundo lanzamiento, y que en los otros dos casos si se ha conseguido.

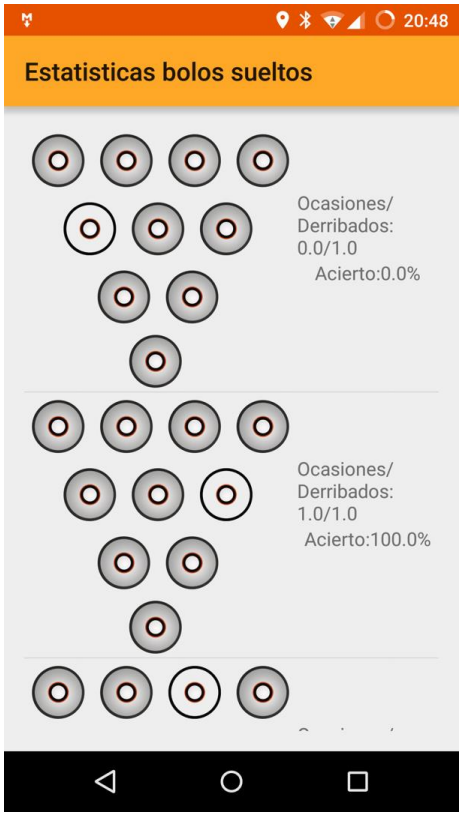

Ilustración 16 Estadísticas bolos sueltos I

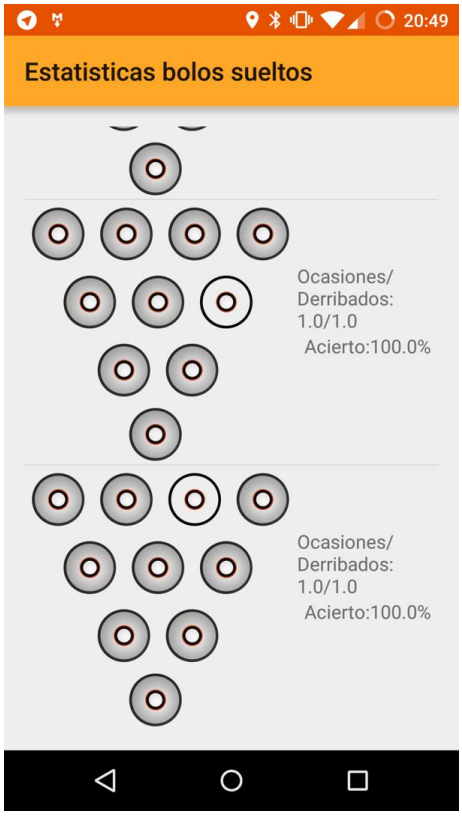

Ilustración 17 Estadísticas bolos sueltos II

Ahora pasamos a las combinaciones multi-bolo o *multipin*, tenemos dos posibles combinaciones en la partida de Tom. La última de ellas no se tiene en cuenta debido a que, al ser el último tiro, nunca se podría llegar a derribar dado que no hay posibilidad de intento.

En la *Ilustración 19 Estadísticas multi-bolo* , podemos observar que tenemos las dos posibles combinaciones. Y como ambas han sido realizadas correctamente en el segundo lanzamiento, obtenemos por tanto el 100% de acierto en ambas durante esta serie de juego de tan sólo una partida.

Ahora pasaremos al resumen. En él tendrá que aparecer una lista con las partidas jugadas, y al pie de la lista tendremos un resumen con las estadísticas más relevantes de la serie.

La puntuación total es correcta. La bola utilizada también.

En el resumen de estadísticas tendremos que el número de partidas en este caso sólo es uno, por lo que también es correcto. En los tantos por cientos tendremos que: el *strike* es correcto dado que tenemos 6/12 lo que es un 50%. En cuanto a los *spare* de bolos sueltos tendremos 2/3 lo que también lo hace correcto. Y en cuanto a *multipin spare* tendremos 2/2 lo que hace que todas las posibles comprobaciones en esta pantalla sean correctas.

Como conclusión de la prueba, se puede decir que ha pasado la validación, cumpliendo con los *Requisitos funcionales* de extracción de datos de la base de datos, y a la vez mostrando estos datos con los valores correctos en *Ilustración 18 Resumen*.

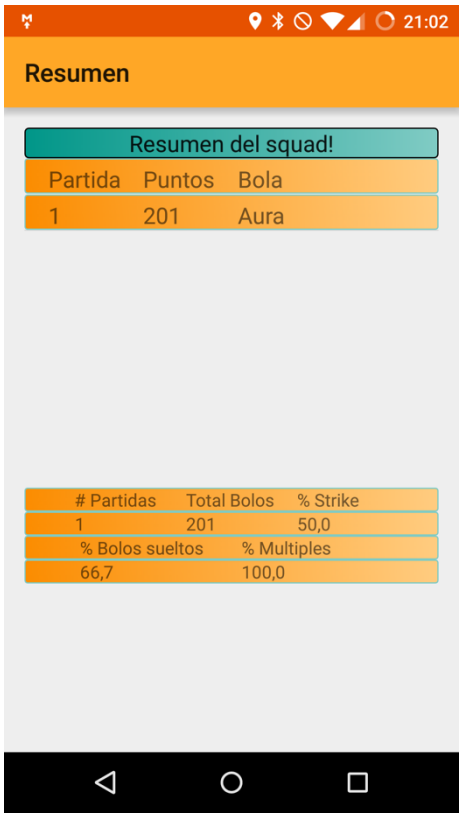

Ilustración 18 Resumen

Ilustración 19 Estadísticas multi-bolo

Estadísticas globales

Ahora pasaremos a hacer una comprobación de las estadísticas de bolos globales (de todas las series jugadas) se accede a través del menú principal. Para ello, partiremos de los siguientes datos:

AMF FRONTIER LANES

7300 EAST THOMAS RD - SCOTTSDALE

(480) 946.5308

1/30/2008 Score 7:15:20PM

Team Team 3					Lane 3				Game 1				1/30/2008
Player	1	2	3	4	5	6	7	8	9	10	Total
BRIAN	- 8	⑦ -	- 8	7 1	- -	8 -	5 2	- 6	7 -	⑧ -	
Hdcp 0	8	15	23	31	31	39	46	52	59	67	67
JOHN	⑧ 1	6 -	9 -	- -	6 2	1 4	8 1	9 -	4 4	8 / 6	
Hdcp 0	9	15	24	24	32	37	46	55	63	79	79
TOM	X	X	7 /	9 /	9 -	X	7 /	X	9 /	X X 7	
Hdcp 0	27	47	66	85	94	114	134	154	174	201	201

Ilustración 20 Ejemplo partidas de bowling

Estas tres partidas se tomarán como partidas de series distintas. Además, tendremos en cuenta que la partida de Tom ya ha sido añadida, lo que provocará que esas estadísticas sean duplicadas.

En primer lugar, como hemos descrito en la anterior validación, iremos a las estadísticas de bolos sueltos. Se puede observar en *Ilustración 21 Estadísticas de bolos sueltos III* y en *Ilustración 22 Estadísticas de bolos sueltos IV* que el bolo 4 no se ha derribado ni una sola vez de las 4 posibles ocasiones. El bolo 6 se ha derribado las 2 ocasiones que ha sido posible, al igual que el bolo 9.

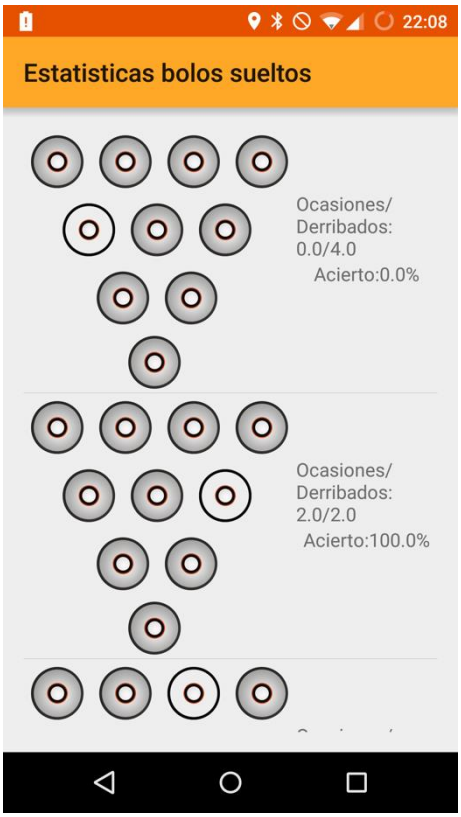

Ilustración 22 Estadísticas de bolos sueltos IV

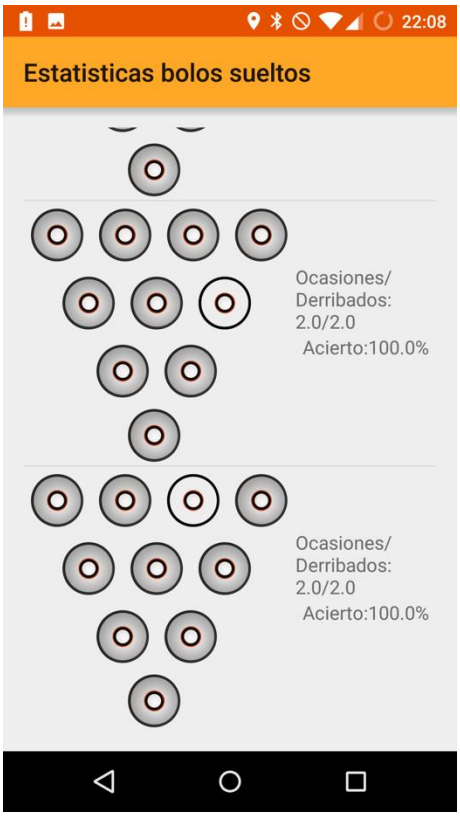

Ilustración 21 Estadísticas de bolos sueltos III

En las estadísticas *multipin* podemos observar las múltiples combinaciones que se han generado. Al ser una cantidad alta de posibilidades, nos centraremos en algunos de los casos.

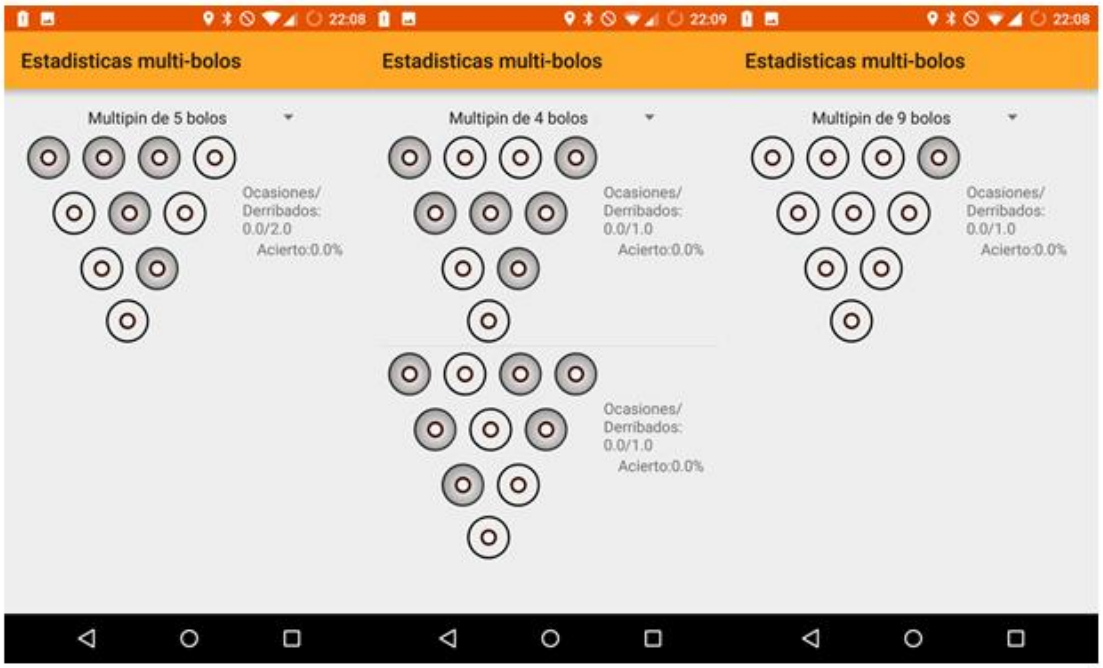

Ilustración 23 Estadísticas de multi-bolo II

En este caso de estudio también se verá que el selector funciona correctamente en los diferentes casos. Éste nos servirá para poder ir filtrando las posibles combinaciones en función de la cantidad de bolos quedados en pie. Y como se puede observar en *Ilustración 23 Estadísticas de multi-bolo II*, los casos son correctos, y se han introducido/extraído correctamente de la base de datos.

4.1.3 Histórico de partidas

En este caso, se comprobará el funcionamiento del histórico de partidas. En él, lo que se espera es ver una lista con todas las posibles series jugadas. Como podemos observar las series jugadas de acuerdo con las partidas anteriores mencionadas, son 4. Como se puede ver en *Ilustración 24 Histórico* el funcionamiento es correcto.

Las fechas y los valores del resumen según los datos extraídos anteriormente también son correctos. Por estos motivos se puede decir que esta prueba pasa correctamente.

Para poder validar el funcionamiento del histórico también tenemos que tener en cuenta que, al pulsar sobre cada una de las series, tenemos que pasar a las estadísticas de ese día. Para comprobar esto, se pulsa sobre una de las series, en este caso la última, para comprobar que nos lleva a la partida de Tom. Y al pulsar sobre el resumen, podremos observar en *Ilustración 25 Resumen* que efectivamente tendremos los datos de ese día. Para poder comprobar en el caso de la partida de Tom que está bien ordenado, introduciremos una variante, la bola.

Para la primera partida, se introducirá la bola llamada "*All Day*" y para el segundo la bola llamada *"gBall"*. Como se puede observar, al hacer *click* en el primer dato del histórico, la bola

mostrada será la llamada "*All Day",* y en el último será la bola llamada *gBall,* por lo que el funcionamiento es correcto.

Ilustración 24 Resumen

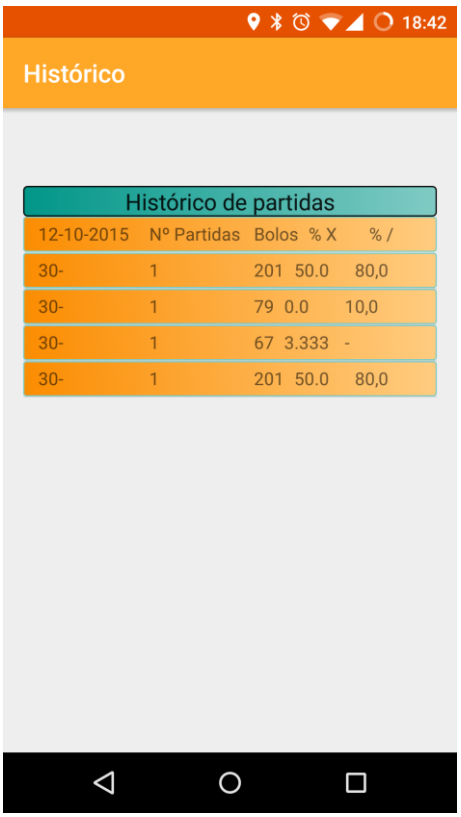

Ilustración 25 Histórico

4.1.4 Carga y guardado de datos de fichero, bola y bolera

Ahora se revisará una parte menos fundamental de la aplicación que es la carga y almacenamiento de los datos de bola y bolera. Estos datos, serán cargados de una lista en formato CSV y serán añadidos en la parte inferior de la aplicación en forma de lista expandible. En estas capturas podremos observar el funcionamiento.

Al pulsar sobre un elemento, la lista desaparecerá y el nombre se añadirá en un color más transparente a la derecha de la etiqueta "Elija bola/Elija bolera". Una vez pulsado el botón "Go Bowling", estos datos serán guardados en base de datos del *squad* siendo mostrados más tarde en la pantalla de resumen. Como podemos ver en *Ilustración 26 Flujo de carga y guardado de datos* el funcionamiento es correcto

Ilustración 26 Flujo de carga y guardado de datos

4.2 VALIDACIONES CON USUARIOS

En este apartado se hablará sobre las validaciones con usuarios potenciales de la aplicación, los cuales fueron sometidos a un breve test, con el objetivo de recoger su opinión sobre la aplicación. Este test incluía además un apartado de posibles mejoras.

Un pequeño grupo de 10 personas realizaron este test, cuyos resultados pueden verse resumidos en *Tabla 1 Validación con usuarios:*

Pregunta/Puntuación	Mejorable	Regular	Buena	Excelente
¿Qué le pareció la apariencia visual?	30%	60%	10%	
¿Le parece útil la app?		30%	70%	
¿Qué le parece la coordinación entre pantallas?			100%	
¿Cubre las necesidades del jugador de *Bowling*?		40%	60%	
¿Qué opina sobre la fluidez de la app?			100%	
¿El funcionamiento		20%	50%	30%

es correcto?				
¿Es la app intuitiva?		40%	50%	10%

Tabla 1 Validación con usuarios

A estos usuarios se les entregó una versión en fase beta, a la cual aún le quedaba desarrollo por delante. Los usuarios además tuvieron un apartado en el que pudieron aportar ideas y opiniones sobre qué mejorar y porqué. Algunas de estas ideas fueron añadidas a la aplicación, pudiendo ser mejorada gracias a estas.

Hay varios apartados en los que los usuarios dan unas notas bajas, las cuales se comentarán a continuación. Se hablará sobre qué se hizo al respecto y porqué.

Los puntos más negativos se obtuvieron en la apariencia de la aplicación. Debido a que no es una aplicación en fase beta, la apariencia no era la prioridad en el momento de las pruebas. Posteriormente, la apariencia se mejoró incluyendo colores del estilo *material desing (*Google Inc 2015) que son los que sugiere Google, además de poner iconos a los botones.

En cuanto al funcionamiento, se obtuvieron también ciertas opiniones negativas. Esto fue debido también a que al ser una fase beta, la experiencia de usuario se vio comprometida con unas funcionalidades aún sin acabar. Estas opiniones vinieron provocadas por unos *crashes* (salida forzosa de la aplicación) debido a búsquedas en base de datos inexistentes, en las cuales no estaban contempladas esta opción.

Algunos usuarios también sugirieron la posibilidad de un servidor en el que poder consultar las estadísticas, o incluso compartir entre usuarios los resultados en tiempo real. Todas estas funcionalidades referentes a la compartición se trasladarán a las líneas futuras de la aplicación. El diseño original de la aplicación no albergará la capa de servidor, quedándose como futuras mejoras.

Las ideas que fueron adaptadas se recogen a continuación.

En primer lugar, al introducir los datos de la partida, todos los bolos “estaban en pie” lo que provocaba que, si hubieras tirado 8 bolos, tendrías que pulsar 8 veces sobre la pantalla para poder hacer la combinación obtenida durante la tirada. Uno de los usuarios sugirió hacerlo de manera inversa, y sólo tener que pulsar sobre los bolos que hubieran quedado en pie, teniendo que hacer menos pulsaciones, y de esta manera ser mucho más cómodo para el usuario.

Otro de los usuarios también sugirió introducir un botón en el área de resumen para poder volver a la HOME una vez terminada la partida. El botón fue introducido, aportando una mejor usabilidad a la aplicación.

Como conclusión en este apartado, se puede decir que la información obtenida por parte de los usuarios es fundamental de cara al desarrollo de una aplicación. Gracias a los *testers* se descubrieron fallos, y además se obtuvo información acerca de cómo se deseaba la aplicación desde el punto de vista de usuarios potenciales. Esta información es de vital importancia para el correcto desarrollo ya que es el usuario final el que le va a dar utilidad.

5 CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se abordarán las conclusiones obtenidas de la realización de este proyecto. También incluirá las líneas futuras que podrá seguir la aplicación una vez finalizado.

5.1 CONCLUSIONES

Con la creación de *BowlingStats* lo que se busca es ofrecer al usuario, un jugador de *bowling* de cualquier nivel, el acceso a una herramienta de trabajo para el análisis de sus estadísticas personales. Para poder alcanzar este objetivo, se toma la idea de aplicaciones ya existentes como: *My Bowling Scoreboard (Ho 2016), Bowling Scorer (Yotchan_lab 2016), Strike Out Stats (Bruno 2016).*

La principal motivación para hacer el proyecto es adquirir la capacidad de poder crear una aplicación móvil desde cero, pasando por todas las fases: diseño, implementación, pruebas, y presentación del producto.

La aplicación maneja una serie de datos guardados en una base de datos con dos tablas como se explica en *Arquitectura*, en los que se guardarán datos tales como: los puntos obtenidos por serie de juego, combinaciones de *spare*, etc. Para manejar estos datos se ha usado el sistema de gestión de base de datos relacional SQLite.

La tecnología escogida fue Android, dado el conocimiento previo en la carrera. Se han conseguido estos objetivos adquiriendo mayor experiencia sobre el terreno.

La aplicación está dirigida para jugadores de *bowling*, para que puedan guardar sus estadísticas y poder estudiarlas, con el fin de mejorar su juego. Esta herramienta puede ser de gran utilidad para el jugador, ya que como en todos los deportes, hay que analizar las estadísticas del juego para saber dónde están las debilidades y fortalezas.

Una vez terminada la implementación una de las conclusiones más importantes es que a la hora de planificar cualquier desarrollo, una de las partes más importantes, es la dedicación de tiempo al diseño de la aplicación. Un buen diseño conlleva a la reducción de costes, ya que se tiene un camino más certero. También es cierto, que actualmente las aplicaciones cambian con el desarrollo, dado que se desarrollan con metodologías ágiles, pudiendo cambiar el producto constantemente, dentro de unos márgenes. Pero una correcta organización es fundamental para el desarrollo de un proyecto exitoso, tanto en implementación, como en tiempo y costes.

5.2 LÍNEAS FUTURAS.

Al finalizar la primera versión de la aplicación *BowlingStats* se empiezan a barajar diferentes opciones por las cuales podría seguir el rumbo de esta aplicación. Las ideas que se han ido recogiendo se plasman a continuación:

- Con el fin de tener una base de datos consistente, se sentiría más cómodo y fiable de cara al usuario poder albergar las estadísticas recogidas en una base de datos en un servidor remoto. Teniendo una base de datos en remoto siendo esta copia de los datos locales, se podría sincronizar con varios dispositivos, y además no se perderían los datos recogidos.

- Gamificación. Al comienzo del proyecto, se proyectó una idea sobre la gamificación de la aplicación, se estudiaron varias posibles opciones. Como *Userinfuser (Userinfuser 2016), Zurmo (Zurmo 2016), CaptainUP (CaptainUP 2016).* Pero el consumo del tiempo durante el desarrollo de las funcionalidades no permitió el desarrollo de esta parte de la gamificación. Por lo que queda como línea futura. Esta funcionalidad haría que los usuarios compitieran entre ellos mediante retos y un sistema de puntos, creando competitividad entre ellos, lo que a la vez provoca un mayor uso de la aplicación.

- Creación de mayor número de datos y estadísticas, como tablas con los promedios a lo largo del tiempo. Datos más técnicos como la introducción de patrones de aceitado, etc.

- Creación de un posible sistema de Alumno-Entrenador, en el que el entrenador, como es obvio, puede tener varios alumnos, y de esta manera poder analizar las estadísticas según las vaya introduciendo el alumno. Con un sistema de mensajería para poder comentar aspectos del juego, y guardar estas mismas notas para poder revisarlas en pasado.

- Referente al sistema Alumno-Profesor mencionado anteriormente, viene unido, un sistema de servidor en el que poder almacenar estas notas del entrenador, para que no se pierdan en caso de cambio de dispositivo.

- Notificaciones para notificar al usuario si el profesor ha decidido hacer algún comentario acerca de las estadísticas, esta funcionalidad es referente a las dos anteriores futuras mejoras.

- Se propone también ampliar el mercado, e implementar la aplicación en Windows Phone e iOS.

ABSTRACT

Today, mobile applications are part of our daily life. We use our mobile phones constantly throughout the day, and therefore thousands of different applications are developed in all possible areas. However, it does not matter how small the field or market is, an application is prepared to meet the different needs. *(PRESS 2016)*

The application developed in the following chapters, is an Android application oriented storage in databases for a subsequent study. This study is aimed to analyze statistical information of a bowler.

In order to do this, you can do a statistical study of the different combinations, scores obtained, percentages of combinations, etc. All at different scales, game series, scores, etc. This study can be made by visualizing the data collected on different lists of statistics.

This application has a local database without going through any server, since the main objective is to store these data on a personal basis, for subsequent study and analysis.

The Android platform was chosen due to the previous usage and development in college and because it is an Open-Source which helps when dealing with possible payment of fees or licenses. The database used is called SQLite, which is from the Operative System Android itself.

When analyzing the data there will be different options. It will be possible to analyze the data available on different days, and within these series of game, the user will be able to analyze the data in those series. As the % of strikes (Knock all the pins down in the first attempt), % Spare (knock down all the pins in two attempts), etc.

The objective of the application is to provide the user with an analysis of its skills as a player in an easy way, yet with a wide variety of data to do further study.

EXTENDED ABSTRACT

CONTEXT

Sir Flinders Petrie, British anthropologist discovered sorts of primitive bowling balls, bowling pins and other materials on Egypt. These bowling pins are from the year 3200 BC. This was accepted and looks like Egyptians play a primitive bowling game 5200 years ago.

The first appearance of British bowling was around 1300. Hard evidence of bowling was found in 1366. In this year, the current king, King Edward III forbid to play bowling, his troops lost a lot of time, and they need to practice more the art of arching. With the King Henry VIII bowling rose again, and became very popular in Great Britain.

In that same period a similar type of bowling was played in the Netherlands. It was the Dutch settlers who introduced the first forms of bowling, which was called 'Dutch pins', in America during the 16th century. In Manhattan the place where the Dutch played this form of bowling is sometimes still referred by as the 'Bowling Green'.

Bowling gets more popular in 1950's when automatic machines appeared, and bowling grew a lot in America, extending this sport around the world. Millions of players practice this game over the years.

Actually, bowling is fighting to become an Olympic Sport. It has millions of players around the world who still continue playing this game. *(Tenpin.org 2016)*

MOTIVATION

The main motivation for the development and choice of this project is the boom that the mobile application market is suffering. From my point of view, to develop a complete application, from designing and development to final stage, is one of the things that motivated me most when choosing this project.

Another major motivation for this project is the need to make a mobile application for tracking statistics for this specific sport, for which, as in many others, it is necessary to manage all possible statistics. When pursuing excellence, these details are very important to improve the technique of the players.

There are already applications on the market for mobile applications (My Bowling Scoreboard *(Ho 2016)*, Bowling Scorer *(Yotchan_lab 2016),* Strike Out Stats (*Bruno 2016)*) the vast majority of payment, because there is not a great market volume, which causes not a lot of competition between applications. This context also serves as motivation for the creation as part of this project a simpler implementation for the user, but with a full utility.

OBJECTIVES

The major objectives when making the application are:

Action plan definition

At this stage, the tools used for development, in order to acquire skills in the technologies chosen and added tools in their development are analyzed. The tools include the use of Git *(Git 2016)* for version control, or use mock ups.

This phase includes the acquisition of the necessary expertise in software development, as Android Studio, Eclipse, using version control software with its subsequent election, programs for designing mobile applications.

Application design

During this phase a design is presented and later modifications are carried according to current standards of design mobile applications. The functional requirements of the application are defined and the architecture needed is created.

Development

In short, the main objective is to obtain knowledge as if it were a startup, addressing all key aspects of development: making decisions based on technology, paths to follow based on certain needs, etc.

The work is the first version of BowlingStats, which covers much of what could be an application of statistics bowling. BowlingStats help bowlers to keep improving his game by analyzing his statistics inserted easily and intuitively.

ARCHITECTURE

The architecture used in the application is an architecture based on insertion into database, and selective extraction of data for displaying it to the user easily to any subsequent findings.

The database used is the database itself Android (SQLite), which provides a set of APIs that give an easy control to the programmer. The tables in the database are:

• Frames

• History

For the required data we will need these two databases according to the requirements defined in the preceding paragraph. Next, the need and the contents of these two tables will be described.

Table Frames:

This table is defined in order to be able to keep every game possible combinations of pins knocked down, so that it can be counted for each bowling pin if it had been standing, or if it had been knocked down.

The decision of how to implement this database was not easy, because had to deal with the fact that, for each frame, there are other 10 chances to contemplate what might have been another style table "Shot" or shooting, which would be met by an identifier, bowling that had been left standing / down, this decision was not made because it would make too many calls to the database, and was thought to be more efficient is another way of working

The table itself contains:

• Squad identifier or series to which it belongs;

• identifier of the item, unique identifier that will make it possible to distinguish between a few games or other;

• The 10 possible frames.

A bowling game has 10 frames, each frame has two possible states, one for each attempt to knock down the bowling pins. This causes two records in the table, one for the first attempt and one for the second attempt to be taken. These states, for simplicity, are defined as follows.

A String with 10 values, zeros and ones, zero symbolizes the bowling pin has been knocked down, and the one, the bowling pin remains. For example, a String '0000000000' symbolizes a strike. In this way, we will have for each frame, the possible combinations that will be relevant for statistical analysis.

On the tenth roll we have the special case, because in this attempt you could throw up to 3 times, so that the database will have 3 possible values for frame 10.

Table History

This table is created for the simplicity of obtaining data on a global level, without having to make calculations in the table itself Frames.

The data contained in this table are:

• identifier of the item

• identifier or serial squad

• Date game

• Ball used

• Lane number

• Total pinFall

• Number of games played in that series.

All this data will give us a summary of each of the series played on any given day, so in one day there could have been disputed different series, which is normal in this type of competition.

SELECTION OF TECHNOLOGIES

When choosing the operating system, you can choose from three widely different operating systems: iOS, Android and Windows Phone. By market share in order they are: Android, iOS and Windows Phone *(Statista.com 2015)*. As for the cost of publication in the market order would be: Android and Windows Phone, iOS. Android and Windows Phone are almost free of charge. The most significant charge comes when the mobile application is uploaded to the market on the other side, iOS, charges an annual fee for each app uploaded to market.

Windows Phone is discarded due to its low market share, also the lack of knowledge of this operating system and low interest to know how it works as a consequence of its low market share.

IOS is discarded because of the hauled costs in their development. For its development it is necessary to purchase a PC of the brand Apple, which are substantially more expensive than the average of Windows computers. What makes interesting the development in this operating system is learning a new operating system, which has currently a high market share. However, this option for the reasons mentioned above is rejected. *(Yarmosh 2015)*

So the operating system chosen for the development is Android, which has already been studied during the previous years of study. In addition, with the ides of not incurring on additional expenses, midrange computer will be sufficient for its development. The upload to the market only involves a single payment.

In conclusion, the Android operating system is chosen because of the previous knowledge, discarding iOS and Windows Phone despite the interest of learning other technologies currently expanding.

REQUIREMENTS

The starting points for the design are the functional requirements of the application, which are:

• The original idea is that the statistics simply are introduced, so that the application was developed in a mobile application, which is currently accessible to anyone;

• interface must be simple so any type of player can simply insert their games statistics;

• statistical model must be scalable to conduct the study on global, or climbing in series of the game, which is what really interests the player;

• The application will allow the show of the various series of game;

• The game series show a short summary of statistics;

• There will not be at any time any connection to the server, everything will be done locally.

In *Ilustración 27 Req. Moqup I* you will see what are the aim of this requirements, you can see the transitions between screens. The first one it’s a moqup off playing one or more games.

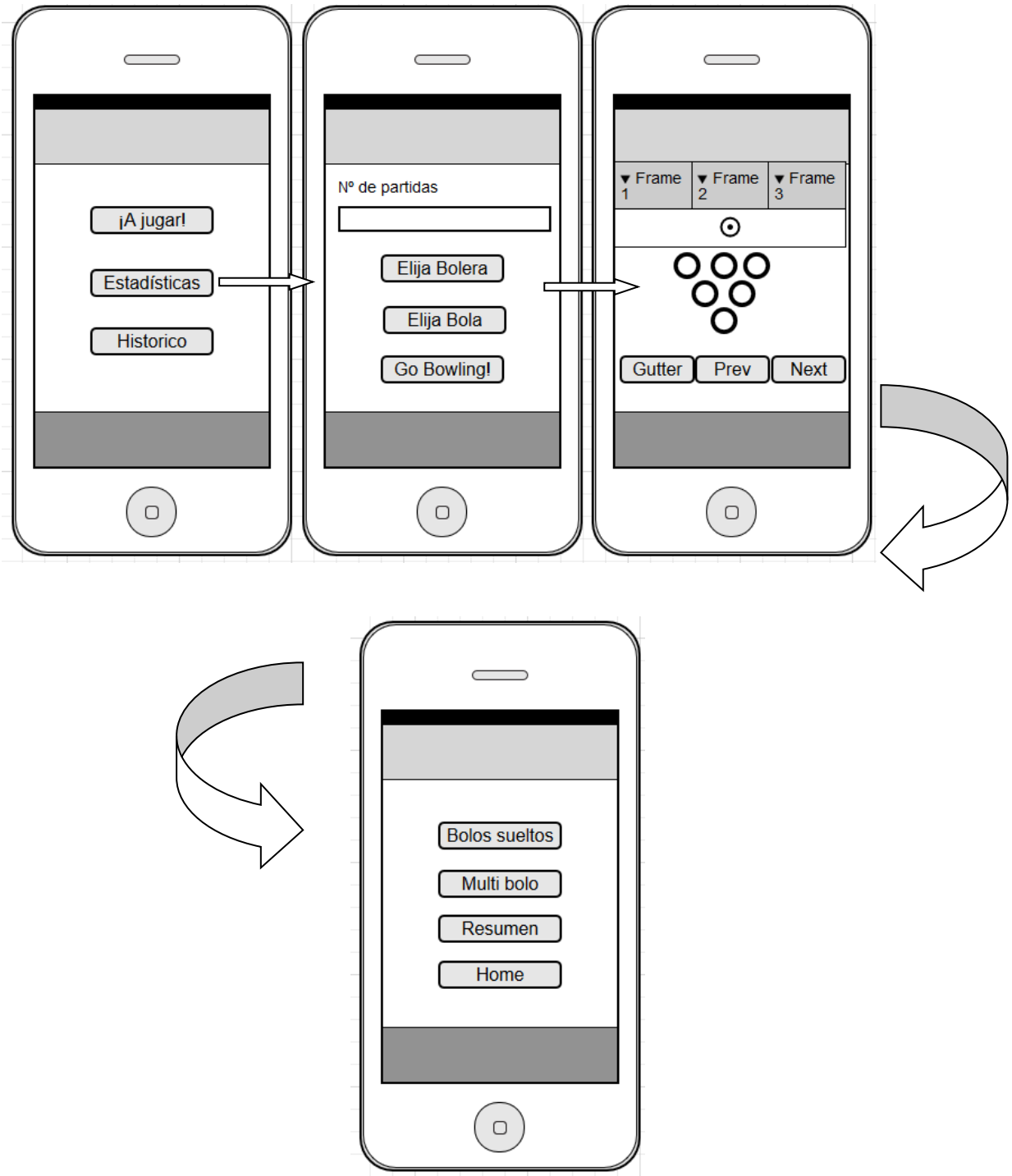

Ilustración 27 Req. Moqup I

You can see the transition between screens, and the flow. You start at Home screen, then if you choose ¡*A jugar!* You will go to another screen with a short questionnaire. When push on *Go Bowling* you enter to the screen where you can insert shot by shot your score. At the end of the game, it will be show another screen with different options like single pin statistics, multi pin statistics, resume, and home button. Pushing on the different options you will go to this action.

In the Ilustración *28 Req. Moqup II*, will be show a different flow between screens. This flow that will be explained in the next lines, it's when single pin and multi pin is pushed, and when resume is pushed.

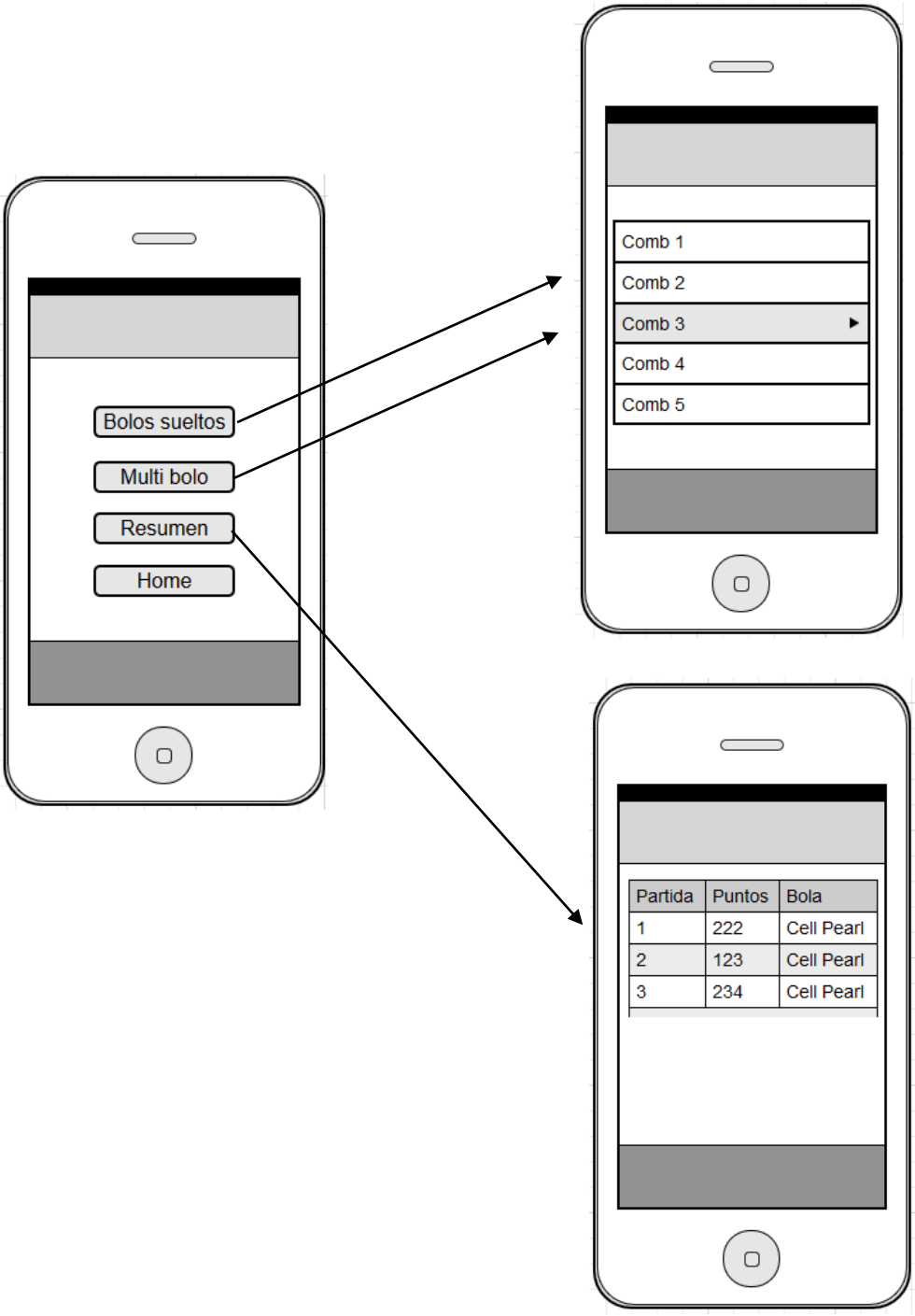

Ilustración 28 Req. Moqup II

The *Ilustración 29 Req. Moqup III* shows when history button is push on, and the transition when one option of the list is clicked. You will go to the screen when you finish the squad, you can choose again the past statistics and the resume of the squad.

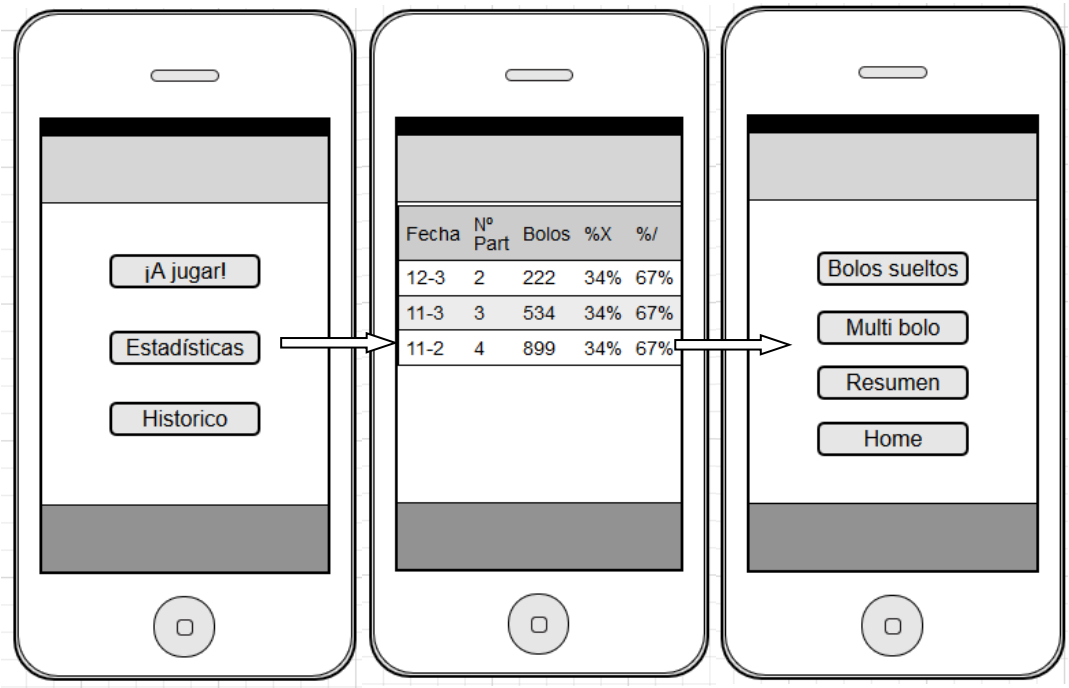

Ilustración 29 Req. Moqup III

IMPLEMENTATION

In this section is discussed in detail the application deployment, developing a detailed outline of the classes used, why this organization, and the need for them. Class diagrams were attached to the explanation, plus graphs to help understanding.

In the diagram below, you can see a brief introduction to possible flows that have the user when interacting with the application.

First, it will present the user with a screen with three buttons: Play, historical, and statistics. The first of these, take the user to a small pre-departure questionnaire, which will have to fill at least the number of games to play in that series. After the questionnaire, we would spend filling in the statistics collected during the game, and later, having finished filling every game, would direct to another screen which would have another 3 buttons: Statistics loose bowling, bowling combinations statistics, summary. In the first two options, as it is said in the requirements, a graph shown with the left combination, and also data on the times we have achieved or not make the semi-full. The third option, the summary will show a short summary of statistics of the series.

Second, the historical will show in list mode, all game series played during our use of the application. Within this list, you can click on each element of it, leading to the screen that would show if we had just had finished playing the series, which would facilitate us to go back to see the most in-depth statistics for that particular case.

Thirdly, we would have the statistics. These concern the pinFall on every possible occasion, we have a comprehensive statistic on all combinations, whether single or combinations of bowling pins. They will be generated again two possible lists, one for single pins and one for multiple bowling pins combinations, separated in different screens.

VALIDATION

As described in Chapter 3, the application has two different areas, the first is the inclusion of the data in the application, and the second display them by lists. These lists cover both statistics collected combinations such as game series played as a brief summary of the data in a series of game. For this reason, we must make a series of checkpoints to ensure that all the data included are correct. To do this, various tests will be carried, both on the insertion of data, and the display of them. All this test passed correctly, a resume of this checkpoints are:

- Insertions on databases is correct.
- Query's to get data from database is correct.
- Score calculate is doing correctly.
- Showing lists are displayed correctly.
- Transitions between screens are correct.

All these validations went successful during the testing operation.

CONCLUSION

With the creation of Bowling Stats what is sought is to offer users a bowler of all levels, access to a working tool for analyzing its personal statistics. To achieve this goal, takes the idea from existing applications as: My Bowling Scoreboard (Ho 2016), Bowling Scorer (Yotchan_lab 2016), Strike Out Stats (Bruno 2016).

The main motivation for the project is to acquire the ability to create a mobile application from scratch, through all phases: design, implementation, testing, and product presentation.

The application handles a number of stored data in a database with two tables as explained in architecture, where data will be stored such as: the number of points earned by playing combinations spare, etc. To manage this data has been used management system relational database SQLite.

Android technology was chosen, given the prior knowledge and study, the costs involved in the development of the application referred to in Choosing the operating system. These objectives have been achieved gaining field experience.

The application is directed to bowlers, so they can keep their statistics and to study them in order to improve your game. This tool can be very useful for the player, because like in all sports, you have to analyze the statistics of the game to know where the strengths and weaknesses are.

After the implementation, one of the most important findings is that when planning any development, one of the most important parts is the time devoted to the design of the

application. Good design leads to cost reduction, since it is a more accurate way. It is also true, that current applications change with development, given that develop agile methodologies may change the product constantly within margins. But proper organization is key when developing a successful project, both in implementation and in time and costs.

FUTURE LINES

After the first version of the application Stats Bowling start shuffling different options which could follow the course of this application. The ideas that have been collected are reflected below:

• In order to have a consistent database, would you feel more comfortable and reliable for the user to hold the statistics collected in a database on a remote server. Having a remote database being this copy of the local data, it could be synchronized with multiple devices, and also data collected are not lost.

• Gamification at the beginning of the project, projected an idea of gamification application, various options were studied. As Userinfuser (Userinfuser n.d.), Zurmo (Zurmo n.d.), CaptainUP (CaptainUP n.d.). But time consumption during the development of the features did not allow the development of this part of the gamification. It remains for future path of growth. This feature would make users compete with each other through challenges and a point system, creating competition between them, which in turn leads to increased use of the application.

• Creating more data and statistics, such as tables with averages over time. more technical and entering data oiling patterns, etc.

• Creation of a possible system of student-coach where the coach, obviously, can have several students, and thus able to analyze the statistics that go entering the pupil. With a messaging system to discuss aspects of the game, and save those notes to review them in the past.

• Concerning the student-teacher system mentioned above, is united, a server system which store these notes from the coach, in order to maintain it and do not lost in case of a change of device.

• Notifications to notify the user if the teacher has decided to make a comment about statistics, this functionality is related to the previous two future improvements.

• It is also proposed to expand the market, and deploy the application on Windows Phone and iOS.

ANEXO A: PLANIFICACIÓN

En la *Tabla 2 Actividades del proyecto* se muestra un resumen de todas las actividades que se han realizado hasta finalizar el proyecto, indicando la fecha de inicio de la actividad y el tiempo que se ha necesitado para su realización.

Actividad	Fecha de inicio	Duración (días)
Documentación del proyecto	29/06/2015	3
Estudio de las tecnologías	02/07/2015	4
Refinamiento del diseño	07/07/2015	2
Programación de la aplicación	07/07/2015	160
Pruebas	13/12/2015	10
Escritura de la memoria	29/01/2016	40
DURACIÓN TOTAL		218

Tabla 2 Actividades del proyecto

A partir de las tareas mostradas en la *Tabla 2 Actividades del proyecto*, se ha creado un diagrama de Gant *(Ilustración 30 Diagrama de Gant*) a partir de las mismas, el cual muestra la evolución con el tiempo de cada tarea.

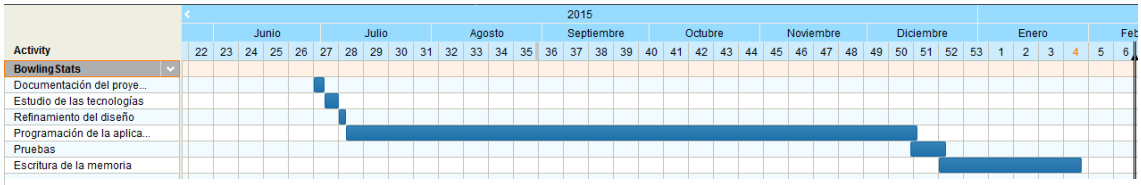

Ilustración 30 Diagrama de Gant

ANEXO B: PRESUPUESTO

En este apartado se realizará la estimación de presupuesto a partir de factores tales como: la cantidad de horas dedicadas al proyecto y los activos necesarios para su realización. El coste de hora por trabajador se ha extraído según los datos del coste mensual de contratación laboral de la Universidad Carlos III de Madrid en el año 2015.

En el proyecto ha participado una única persona en el desarrollo, titulado medio (10€/hora), y dos titulados doctores al 2% de la jornada (30€/hora), con el mismo salario. Por lo que según las horas totales que se extraen de la tabla anterior tenemos que los gastos totales de personal ascienden a 31.098€,

Número de actividad	**Definición**	Días utilizados	Horas/Titulado medio	Horas/Titulado profesor
1	Documentación del proyecto	3	12	0.48
2	Estudio de las tecnologías	4	16	0.64
3	Refinamiento del diseño	2	8	0.32
4	Programación de la aplicación	160	640	25,6
5	Pruebas	10	40	1.6
6	Escritura de la memoria	40	160	6.4
TOTAL DE DURACION		219	876	35

Tabla 3 Actividades del proyecto con el tiempo dedicado

PRECIO POR HORA	TOTAL DE HORAS	PRECIO TOTAL PERSONAL (€)
10	876	8.760
30	35	1.050
30	35	1.050
COSTE TOTAL PERSONAL		10.862

Tabla 4 Presupuesto del personal

En la *Tabla 5 Coste del material*Tabla 4 Presupuesto del personal se realiza una estimación del presupuesto necesario en cuanto a material se refiere. Para poder hacer esta estimación tenemos que para desarrollar la aplicación en cuestión se necesita: un ordenador de gama media, y el alquiler de un local para su realización. Por lo que:

MATERIAL	**PRECIO**	**COSTE (€)**
Ordenador gama media	900 €	200
Local	200€/MES	1400
COSTE TOTAL DE MATERIAL		1600

Tabla 5 Coste del material

En la *Tabla 6 Presupuesto total del* proyecto se muestra el presupuesto total para la realización del proyecto.

COSTES	**VALOR EN €**
PERSONAL	10.862
MATERIAL	1600
SUBTOTAL	12.462
I.V.A (21%)	2.617
TOTAL	15.079

Tabla 6 Presupuesto total del proyecto

ANEXO C: MARCO LEGAL

Al crear aplicaciones móviles, al igual que cuando creas cualquier producto, hay una serie de problemas legales que pueden llegar a surgir. En el caso de las aplicaciones móviles, hay una regulación vigente tratada por los organismos correspondientes.

Estas regulaciones vienen de diferentes escalas tales como el dictamen conjunto sobre la privacidad en las aplicaciones móviles de marzo de 2013, el cual se refiere a la necesidad de la protección de los derechos de los usuarios. Esta ley, afecta a todo el conjunto, tanto al creador o desarrollador, al distribuidor, y al usuario o cliente encargado de dar uso a la misma. Una de las partes más relevantes de este dictamen es la necesidad del consentimiento por parte del usuario. Además del consentimiento, el usuario tendrá derecho a la modificación y/o eliminación de los datos entregados. También cabe destacar lo dispuesto en el dictamen sobre la responsabilidad de las distintas personas que intervienen a lo largo del proceso de las aplicaciones móviles, estableciendo cual es la responsabilidad de cada una de ellas respecto a los usuarios.

También cabe destacar que, junto a este dictamen a nivel europeo, se debe considerar la Directiva de Datos95/46, como la Directiva 2002/58/CE de Privacidad y Comunicaciones Electrónicas. En estos documentos es donde se hace la directiva para la necesidad del consentimiento previo del usuario.

A nivel nacional, se puede destacar la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico y la Ley Orgánica de Protección de Datos de Carácter Personal. En la primera se indica la necesidad de establecer un informe con la información personal del responsable de la aplicación. Además, en esta ley se regula el tema de publicidad dentro de las propias aplicaciones. En la Ley Orgánica de Protección de Datos de Carácter Personal se hacen referencia a muchos derechos del usuario entre los que Ilustración n el derecho de acceso, rectificación, cancelación e indemnización.

BIBLIOGRAFÍA

1. Bruno, William. *Google Play.* 2016 https://play.google.com/store/apps/details?id=com.bruno.myapps.strikeoutstats.
2. CaptainUP. 2016 https://captainup.com/.
3. Chapin, Tom. *Bowling Score.* 2016 http://www.tchapin.com/?attachment_id=108 (último acceso: Enero de 2016).
4. Corporation, Mozilla. 2016 https://developer.mozilla.org/es/Firefox_OS (último acceso: Noviembre de 2015).
5. Git. *Git Documentation.* 2016 https://git-scm.com/doc (último acceso: Noviembre de 2016).
6. Ho, Peter. *Google Play.* 2016 https://play.google.com/store/apps/details?id=com.peterhohsy.mybowling.
7. Inc, Google. *Material Desing.* 2016 https://www.google.com/design/spec/material-design/introduction.html.
8. Michelone, Manuel López. *La historia de Android.* 2016 http://www.unocero.com/2013/09/23/la-historia-de-android/ (último acceso: Septiembre de 2015).
9. PRESS, EUROPA. *elEconomista.es.* 5 de 1 de 2016. http://www.eleconomista.es/apps/noticias/7260042/01/16/El-uso-de-aplicaciones-moviles-sube-un-58-en-2015-segun-Flurry-Yahoo.html.
10. RW. *La historia del sistema iOS de Apple, en imágenes.* 2016 http://www.reasonwhy.es/actualidad/tecnologia/la-historia-del-ios-de-apple-en-imagenes (último acceso: Septiembre de 2015).
11. Sportsvite. *sportsvite.* 2016 http://sportsvite.com/sports/Bowling/rules.
12. Statista.com. *Statista.com.* 2015 (http://www.statista.com/graphic/1/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system.jpg) (último acceso: Septiembre de 2015).
13. Userinfuser. 2016 https://code.google.com/archive/p/userinfuser/.
14. Wikipedia. 2015 https://es.wikipedia.org/wiki/Ubuntu_Touch (último acceso: Septiembre de 2015).
15. *Windows Phone.* 2016 http://www.elotrolado.net/wiki/Windows_Phone (último acceso: Septiembre de 2015).
16. Yarmosh, Ken. *Android vs iOS: Which platform to build for first?* 2015 http://savvyapps.com/blog/android-vs-ios-which-platform-to-build-for-first (último acceso: Septiembre de 2015).
17. Yotchan_lab. *Google Play.* 2016 https://play.google.com/store/apps/details?id=jp.gr.java_conf.yotchan.bowlingscorerfree.

18. Zurmo. 2016 http://zurmo.org/.

19. Sgoliver. Blog personal de tutoriales de Android. (http://www.sgoliver.net) (último acceso: diciembre 2015)
20. Coursera. Web didáctica dedicada a la impartición de cursos online. (www.cousera.com) (último acceso: Septiembre 2015).

21. Udemy.Web didáctica dedicada a la impartición de cursos online. (http://www.udemy.com) (último acceso: Septiembre 2015).

22. StackOverflow. Foro dedicado a la consulta y resolución de dudas y errores de programación. (http://www.stackoverflow.com) (último acceso: Diciembre 2015).

23. MySQL. Panorámica del sistema de gestion de bases de datos MySQL. (http://dev.mysql.com/doc/refman/5.0/es/what-is.html) (último acceso: Noviembre 2015).

24. Tomsplanner. Gestión de proyectos online (Creación del diagrama de Gant). (https://www.tomsplanner.es) (último acceso: Enero 2015).